O Projeto PEGASUS/PLURIX/TROPIX, um UNIX brasileiro

Pedro Salenbauch (editor, autor), Newton Faller (in memoriam), et al.
Instituto Tércio Pacitti – NCE
Universidade Federal do Rio de Janeiro - UFRJ

1. Introdução AO PLURIX

Durante os anos de 1983 a 2012, o NCE/UFRJ desenvolveu o projeto PEGASUS/PLURIX/TROPIX, e este artigo pretende descrever o histórico deste empreendimento.

A partir do ano de 1976, diversos pesquisadores do NCE/UFRJ encontravam-se nos Estados Unidos e na Europa em cursos de doutoramento, época em que o Sistema Operacional UNIX começou a ser instalado no computadores das Universidades em que eles se encontravam. O UNIX é um Sistema Operacional desenvolvido nos Laboratórios Bell da AT&T, cuja filosofia, na época revolucionária, influenciou o desenvolvimento de inúmeros Sistemas Operacionais.

Durante esta época, inúmeras teses foram desenvolvidas em ambientes UNIX, gerando sistemas específicos para execução em sistemas UNIX. Estes pesquisadores naturalmente tornaram-se apreciadores do Sistema. No entanto, ao retornarem ao Brasil, a partir de 1979, depararam-se com a ausência deste Sistema no Brasil. Foi então tentado, através do NCE/UFRJ o licenciamento do Sistema diretamente da AT&T. As negociações foram iniciadas, mas infelizmente, as respostas da AT&T aos pedidos do NCE/UFRJ foram sempre ambíguas e a decisão final foi sendo sempre adiada e o licenciamento jamais acabou-se concretizando. O motivo foi que o Brasil na época era considerado um país “pirata”, que não respeitava os contratos de licenciamento.

Nos anos subsequentes, ainda mais pesquisadores que haviam apreciado o uso do UNIX no exterior retornaram ao Brasil. Com isto, aumentava o número de pesquisadores descontentes, pois no NCE/UFRJ não havia computadores com UNIX, e o que é pior, não havia sequer a perspectiva de tê-los a curto prazo. Uma decisão impunha-se na época: ou desistia-se de uma vez por todas de ter ambientes computacionais no estilo UNIX no NCE/UFRJ, ou tentava-se um desenvolvimento próprio. Decidiu-se pela segunda alternativa.

O desenvolvimento de um novo Sistema Operacional, embora possa parecer inviável à primeira vista, na realidade não o é. Em primeiro lugar, o UNIX foi originalmente desenvolvido por apenas dois pesquisadores em um prazo de dois a três anos. Estes dois pesquisadores definiram e implementaram o Sistema. No nosso caso, o UNIX já estava definido, e confiávamos em nossa experiência anterior, pois já havíamos implementado Sistemas Operacionais para Microcomputadores. Além disto, os pesquisadores do NCE/UFRJ já haviam trabalhado

extensivamente com o UNIX nas Universidades do exterior e já havia muita documentação disponível. O desenvolvimento iniciou-se em 1982.

O desenvolvimento de um Sistema Operacional é uma tarefa trabalhosa e complexa, e para que possa chegar a bom termo em um período de tempo aceitável é essencial um domínio total sobre o ambiente de desenvolvimento. Como a interação entre um Sistema Operacional e o Computador é muito grande, o controle do ambiente de desenvolvimento se estende também ao Computador ("Hardware").

Para que este total controle se efetivasse, foi desenvolvido em paralelo o PEGASUS–32x, um Supermicro multiprocessado baseado na família de microprocessadores 680x0 da Motorola. O desenvovimento do PLURIX está tão intimamente relacionado ao desenvolvimento do PEGASUS–32x, que nos é difícil imaginar a possibilidade de desenvolver-se um sem o outro.

Como já dito, o desenvolvimento de um Sistema Operacional é uma tarefa trabalhosa e complexa, e além disto, uma tarefa cara. Os recursos computacionais e, principalmente, humanos necessários para tal desenvolvimento são altamente especializados e, conseqüentemente, os custos são elevados. Estes custos do desenvolvimento do PLURIX foram em quase a sua totalidade cobertos por recursos do próprio NCE/UFRJ, mas houve uma complementação através de uma verba do FIPEC do Banco do Brasil. Esta verba foi solicitada no início de 1984, depois de muitas explicações e formulários e foi recebida quase dois anos mais tarde. Embora de valor pequeno se comparado ao valor total do projeto, este auxílio do FIPEC veio permitir a compra de alguns periféricos essenciais para o desenvolvimento do Sistema.

Ainda sem verbas e sofrendo muitas críticas por causa disto, o projeto chegou ao fim de 1984 com um protótipo básico em funcionamento. O ano de 1984 foi assim, crucial para o desenvolvimento do PLURIX. Neste ano ficou demonstrada a viabilidade do seu desenvolvimento, pelo menos para os pesquisadores envolvidos no projeto, podendo constituir-se em um produto comercial em um futuro próximo.

A coordenação do projeto, apesar da descrença generalizada, sugeriu até o licenciamento do Sistema para as indústrias nacionais que se interessassem por um sistema alternativo ao UNIX. Um embrião deste sistema foi apresentado na Feira de Informática de 1984, embora sem despertar grandes interesses.

Durante o ano de 1985 e principalmente em 1986, depois do recebimento das verbas, diversas pessoas foram contratadas para trabalhar no projeto. O PLURIX pôde ser expandido de forma a refletir o padrão do UNIX System V e um grande número de utilitários e rotinas foram escritos. Com mais de 70 utilitários, a Versão 1.0 já se constituía em um excelente ambiente para desenvolvimento de "software" possuindo desde editores e processadores de texto até diversas sofisticadas ferramentas apreciadas pelos conhecedores do ambiente UNIX.

Desde o início de 1985 o Sistema vinha sendo homologado. Isto porque, como filosofia do projeto desde que o Sistema atingiu condições mínimas de uso, ele tem sido utilizado para o seu próprio desenvolvimento. Assim, cada nova rotina ou utilitário que ia sendo desenvolvido e incorporado ao Sistema era extensivamente testado por todos os membros da equipe de desenvolvimento através da sua constante utilização. A garantia, portanto, de que o sistema se constituiu em um bom

ambiente de “software” e era justamente o fato de ele próprio ter sido desenvolvido em tal ambiente.

Muitas pessoas estiveram envolvidas no desenvolvimento do PLURIX durante as suas diversas fases. O grande incentivador do projeto no seu início foi o analista Prof. Luiz Antonio da Cunha Couceiro, também um grande apreciador de sistemas com filosofia UNIX. A concepção do projeto e a idéia do seu licenciamento são devidas ao analista Prof. Newton Faller que exerceu a coordenação geral do projeto. É importante salientar o seu esforço no sentido de divulgar o PLURIX também fora do ambiente acadêmico, o que despertou a atenção da indústria para o inaproveitado potencial da Universidade e em particular, do NCE/UFRJ. A coordenação do desenvolvimento do PLURIX foi exercida pelo analista Prof. Pedro Salenbauch que, com certeza, é a pessoa que mais contribuiu para o desenvolvimento do sistema. No ano de 1986 a equipe de desenvolvimento do PLURIX era formada pela analista Eliana Barros de Oliveira Paiva, e pelos estagiários Gustavo Peixoto de Azevedo, Haroldo Ricardo Juvenal de Macedo, Júlio Tadeu Carvalho da Silveira, Oswaldo Vernet de Souza Pires, Rafael Peixoto de Azevedo e Sílvia Maria de Andrade Barbosa.

As pessoas envolvidas no projeto PEGASUS e que contribuíram para o desenvolvimento do PLURIX foram os analistas Manuel Lois Anido, Gabriel Pereira da Silva, José Luiz Ribeiro Filho, Norival Ribeiro Figueira e Luiz Fernando Huet de Bacellar, e os estagiários Alexandre Malheiros Meslin e Paula Tavares da Cunha Melo. Em diversos estágios do projeto outras pessoas também contribuíram para o desenvolvimento do PLURIX. São eles os analistas Chun Yin Hsu, Gonzalo Archondo Callao, Carlos Alberto Perez Muiños, Carlos Eduardo Mendes de Azevedo e os estagiários Alexandre Botão e Álvaro da Silva Lima Filho.

O PLURIX, registrado na SEI em agosto de 1986 na categoria A, teve a sua versão 1.0 concluída em março de 1987. Na época, já tinha sido licenciado pela EBC (Empresa Brasileira de Computadores e Sistemas S. A.) e estavam em tramitação na UFRJ os pedidos de licenciamento pela LOGUS, ITAUTEC, CMW e MICROTEC. Esta versão era composta do núcleo do Sistema, utilitários e rotinas da biblioteca, conforme definido na publicação “System V Interface Definition” da AT&T como “sistema básico”. A Versão 1.0 contém ainda inúmeras extensões que foram julgadas úteis para o desenvolvimento de “software” e que não são encontradas na definição do UNIX. Essas extensões vêm sendo criadas naturalmente pelos projetistas do PLURIX com o objetivo de facilitar o seu próprio trabalho de desenvolvimento já que o PLURIX em execução no PEGASUS tem sido utilizado para o seu próprio desenvolvimento.

A Versão 2.0 de 1988, além das facilidades já oferecidas na Versão 1.0, incluiu um compilador para a linguagem “C”, um interpretador de comandos (“shell”), bem mais sofisticado do que o da Versão 1.0, e facilidades para o tratamento de múltiplas janelas, permitindo o desenvolvimento de aplicações de utilizam ícones e “mouse”.

Nesta época, a Versão 3.0, prevista para 1989, permitiria a implementação de um ambiente distribuído composto de vários PEGASUS (ou outros computadores). O usuário poderia trabalhar neste ambiente de forma transparente, isto é, como se estivesse trabalhando em um computador local, sem qualquer preocupação com comunicações. A completa definição da Versão 3.0 somente seria concluída quando fossem estabilizadas, no exterior, as diversas correntes que propunham, cada uma delas, um padrão de protocolos diferente. Aguardávamos, por exemplo, uma melhor

definição internacional do padrão ISO/OSI (International Standards Organization/Open Systems Interconnection) aos ambientes com filosofia UNIX antes de iniciarmos o seu desenvolvimento.

Nesta época a padronização do UNIX estava saindo do controle da AT&T e indo para o IEEE (Institute of Electrical and Electronics Engineers) através do POSIX (Portable Operating System UNIX style), o padrão de comunicações para o UNIX que ainda estava sendo definido, etc ...

2. Características do PLURIX

O sistema PLURIX baseia-se nas especificações do Sistema UNIX, possuindo diversas melhorias encontradas em outros sistemas, além de características próprias.

Entre as principais características do PLURIX podemos citar:

1. Suporte para o controle de múltiplos processadores.
2. Sistema de arquivos hieráquico, incluindo volumes montáveis.
3. Entrada/saída em arquivos, em dispositivos e entre processos compatíveis entre si.
4. Ativação de processos assíncronos.
5. Seleção do interpretador de comandos do sistema a nível de usuário/aplicação.
6. Alto grau de portabilidade, sendo a grande maioria do sistema escrito em uma linguagem de alto nível (“C”).
7. Chamadas às funções do sistema (“System Calls”) padronizadas.

O Sistema de Arquivos possui cinco tipos de arquivos: regulares, diretórios, especiais de blocos, especiais de caracteres e “fifos”.

Um arquivo regular contém qualquer informação nele colocada pelo usuário. Para o sistema, estes arquivos não assumem nenhuma estruturação, e são simplesmente uma seqüência contínua de bytes. O acesso a estes arquivos pode ser sequencial ou direto, endereçado a qualquer byte do arquivo.

Os diretórios implementam o mapeamento entre o nome do arquivo e o arquivo propriamente dito, e assim descrevem a estrutura do sistema de arquivos como um todo (em forma de árvore). Um diretório pode ter um elo (“link”) para qualquer tipo de arquivo. Qualquer arquivo (exceto um diretório) pode ser apontado por mais de um elo. Existe um diretório especial, reconhecido pelo sistema, chamado de raiz (“root”, a raiz da árvore), a partir do qual pode ser encontrado qualquer arquivo do sistema de arquivos, bastando para tal, especificar a seqüência de diretórios a ser consultada.

A cada dispositivo físico de entrada/saída (disco, fita, linha de comunicação, etc.) está associado um arquivo especial. O acesso a estes arquivos especiais através de uma operação de entrada/saída causa, na realidade, a ativação física do dispositivo associado. No caso de dispositivos de disco, é possível dividí-los em várias unidades lógicas diferentes (partições) e neste caso teremos além do arquivo especial para o disco como um todo, um arquivo especial para cada unidade lógica.

Todos os dispositivos físicos possuem um arquivo especial de caracteres associado. Além disto, os discos, que normalmente contém imagens do alguns blocos na memória interna para aumentar o desempenho do sistema (“cache”) possuem arquivos especiais de blocos associados. Através disto, podemos realizar operações de entrada/saída de/para estes blocos da memória interna.

Um arquivo tipo “fifo” são utilizados para comunicações entre processos (ver adiante). Para tal, o processo leitor deve abrir um arquivo fifo para leitura, enquanto que o processo escritor deve abrir o mesmo arquivo para escrita. Assim, qualquer informação escrita por um dos processos será repassada diretamente ao outro (na realidade este tipo de arquivo é um “pipe” com nome, ver adiante).

Um processo é a execução da imagem de um programa, o ambiente de execução do computador. A imagem consiste basicamente de duas partes: a primeira contém várias informações necessárias para a execução do processo, tais como os valores dos registros do processador, registros de gerência da memória, os arquivos abertos, a identificações do usuário, etc... A segunda contém a área do texto do programa sendo executado (o código objeto), a sua área de dados e a pilha (“stack”) do processo.

Um processo pode criar um outro processo assíncrono através da chamada ao sistema “fork”. Este novo processo será uma cópia idêntica do processo antigo, isto é, a imagem dos dois processos será igual. A diferença entre os dois será o valor retornado pela chamada ao sistema. O processo inicial (“pai”) ao retornar, receberá um número único identificando o novo processo criado (processo “filho”), enquanto que este receberá o valor zero.

Um processo pode substituir sua área de texto, dados e pilha através da chamada ao sistema “exec”. Esta chamada recebe como argumento o nome de um programa a ser executado e os seus parâmetros. Assim, todo o texto e dados do processo serão substituídos pelo conteúdo do arquivo, mas a primeira parte da imagem não é alterada. É bom observar que a chamada “exec” tem como objetivo executar este novo programa, e nunca retorna, exceto em caso de erro.

Um processo pode esperar o fim da execução de um dos seus processos filhos através da chamada ao sistema “wait”. Esta chamada retorna a identificação do processo filho que terminou, assim como o estado da terminação, isto é, a informação se ocorreu algum erro durante a execução do processo filho.

Para a comunicação entre dois processos “irmãos” existe um mecanismo chamado de “pipe”. O “pipe” é um canal de comunicação similar ao “fifo”, que permite a um processo escrever informações no “pipe”, que serão lidas pelo outro. Para utilizar um “pipe”, o processo deve criá-lo através da chamada ao sistema “pipe”, e ele será passado ao processo filho através da chamada ao sistema “fork”, já que um “pipe” é tratado como um arquivo comum.

Em relação ao controle de entrada/saída, os pedidos de entrada/saída dos usuários passam por uma interface usuário/sistema operacional, sendo analisados e distribuídos a funções específicas de entrada/saída (acionadores) para os diversos periféricos do sistema.

Há dois tipos básicos de funções de entrada/saída: as “estruturadas” (ou “de bloco”), que tratam periféricos tais como discos, cujos blocos são processados através de cópias na memória interna (“cache”, para aumentar o desempenho) e as “não estruturadas” (ou “de caracteres”), que tratam

de periféricos tais como terminais e impressoras. Repare que os discos também podem processados através das funções “de caracteres”, quando desejamos acessá-los diretamente (sem passar pelas cópias dos blocos residentes na memória interna).

As funções “de blocos” manuseiam um conjunto de áreas de tamanho fixo, que contêm os dados. Através de “cabeças de áreas” encadeadas, são mantidas listas de áreas associados a cada periférico “de blocos”, e uma lista de áreas disponíveis.

As funções “de caracteres” também manuseiam filas formadas pelo encadeamento de pequenos blocos, alocados e desalocados quando necessário, que contêm os caracteres. Isto é feito para terminais e impressoras, como exemplo. Devido à existência de Unidades de Processamento Periférico no PEGASUS-32x, o menuseio de caracteres é feito de maneira descentralizada, liberando as UCPs para funções mais importantes.

Os diversos periférico são identificados por dois códigos: um seleciona o seu acionador, através de uma entrada correspondente em uma tabela de configuração do sistema. O outro é repassado ao acionador selecionado, para escolher a unidade de um controlador, ou a partição de um disco, por exemplo.

Os processos no PLURIX podem estar basicamente em três estados: RUN, READY e SLEEP. Um processo em estado RUN está sendo executado em algum dos processadores do PEGASUS-32x. Os processos no estado READY estão prontos para serem executados e aguardam um processador livre. Em SLEEP, um processo está aguardando um recurso que está sendo utilizado por outro processo (mas que não é um processador).

Os processo no estado READY são organizados em filas ordenadas por prioridade. Garante-se que em qualquer instante de tempo, os “n” processos prontos para rodar, de maior prioridade, estarão sendo executados nos “n” processadores disponíveis. Isto só não ocorrerá no caso em que for definido que determinado processo só pode ser executado em certo processador (normalmente um processo pode ser executado indistintamente em qualquer dos processadores disponíveis).

O balanceamento da carga de trabalho é feito através da mudança dinâmica das prioridades à medida em que os processos vão sendo executados. Um esquema de realimentação negativa garante a estabilidade e eficiência do sistema.

O esquema adotado para o multiprocessamento é o de total simetria em relação às UCPs. Desta forma qualquer trecho de código, seja de usuário ou do supervisor por ser executado em qualquer das UCPs disponíveis. Por esta razão especial, atenção foi dada aos recursos que exigem exclusão mútua.

Um esquema de semáforos, que em seu nível mais baixo está baseado nas instruções de “test-and-set” dos processadores, garantem a correta utilização de recursos mutuamente exclusivos. Especial atenção tem sido dada à instrumentação do sistema de forma a localizar e corrigir eventuais “dead locks” (veja abaixo o capítulo sobre o Sincronismo do PLURIX).

Em relação à interface entre o sistema e o usuário, o PLURIX apresenta os mesmos comandos e utilitários básicos do UNIX, além de suas principais opções. O principal interpretador de comandos

(“shell”), possui as mesmas características básicas, que são: execução de programas com parâmetros; redirecionamento de entrada/saída; execução de arquivos de comandos.

A inicialização do sistema inicia-se com o seu auto-teste. Em seguida, a UCP0 executa a função de carga, que se encontra em ROM, enquanto as outras UCPs esperam uma mensagem da UCP0. A função de carga é bastante inteligente, no sentido de poder efetuar a carga de qualquer dispositivo do sistema (discos rígidos, disquetes, etc ...). Essa função também reconhece o sistema de arquivos do PLURIX, podendo carregar qualquer programa que se encontre nos dispositivos do sistema. Para tal, a função espera que o operador tecle na console o nome do arquivo e o periférico aonde este se encontra.

Carregado o sistema operacional, este inicializa suas tabelas, cria o processo 0 (o escalador de processos) e o diversos expedidores (processos 1 até “n”, onde “n” é o número de UCPs). Em seguida, é criado o processo “init” que será o pai de todos os processos subsquentes. Ele criará um processo para cada terminal, que em seguida executará o programa “login”, passando o controle para um “shell” quando o usuário for admitido no sistema.

3. O PEGASUS-32x

Para viabilizar a implementação do Sistema Operacional PLURIX, era necessária a existência de um computador no qual o PLURIX iria executar. Por isto, foi desenvolvido concomitantemente com o PLURIX, um novo supermicrocomputador, denominado de PEGASUS-32x.

O PEGASUS-32X representa uma família de supermicrocomputadores, homogêneos, simétricos, de 32 bits, construídos com diversas unidades de processamento (UCPs) da famíla Motorola MC680x0 operando em paralelo (Multiprocessamento), Unidades de Processamento Periféricos (UPPs) inteligentes para entrada e saída (E/S), memória global e barramento de intercomunicação com um barramento VME (Versa Module Europa bus).

Na fase de concepção, levou-se em consideração de que a construção de sistemas de multiprocessamento homogêneos (aqueles em que os módulos básicos como UCPs, Unidades de Memória (UMs) e Unidades de Processamento Periferico (UPPs) podem ser compartilhadas indistintamente, apresentam uma série de vantagens sobre outros, tais como:

1. O aumento do desempenho (porte) do sistema é imediato, já que ele é composto de módulos homogêneos e, portanto, basta replicá-los.
2. Quanto mais homogêneo, mais robusto (menos sujeito a falhas), pois ainda restam recursos semelhantes, quando um falha.
3. A utilização de um módulo básico em cada parte do sistema multiprocessador diminui os custos de projeto, fabricação e programação.
4. A homogeneidade propicia uma só versão do “software”, independente do porte do Sistema.
5. O tempo de desenvolvimento é muito menor, pois o desempenho não está relacionado à complexidade dos módulos, mas sim à sua quantidade.

A simetria dos módulos do sistema permite uma melhor utilização dos recursos computacionais, tanto em operações de cálculo, com em entrada/saida.

Todas as UCPs, Unidades de Processamento Periférico (UPPs) e Unidades de Memória (UMs) são interconectadas pelo barramento global do Sistema (VME Bus). Todas as UPPs e UMs são endereçadas de forma idêntica por todas as UCPs. Isto significa que um programa pode ser executado em qualquer um dos processadores, sem qualquer modificação no modo de endereçar a memória ou as Unidades de Processamento Periférico. Da mesma forma, todas as UPPs possuem uma visão simétricas das UCPs e da memória.

Esta simetria é também explorada para melhorar a confiabilidade do sistema. Quando o mesmo é ligado, cada módulo realiza um auto-teste, e reporta o resultado a um dos processadores que é designado como mestre. Esta UCP mestre inicia a execução do código do Sistema Operacional, que determina o número de componentes do Sistema que passaram no auto-teste e configura o Sistema, baseado nos componentes operacionais.

Desde o início do projeto, deu-se uma atenção especial à capacidade e facilidade de auto-teste de cada um dos módulos do sistema. Isto provou ser de enorme valia, pois os testes dos módulos foram feitos em paralelo independentemente, concorrendo para o rápido desenvolvimento do sistema.

O Objetivo foi construir uma máquina de 32 bits, utilizando os microprocessadores da época de 16/32 bits, que pudesse competir em desempenho com o VAX-780 ou o IBM 4341-MG2, a custo inferior e que utilizasse o Sistema Operacional PLURIX. Esta máquina deveria possuir também uma elevada capacidade de auto-teste e replicabilidade.

As características gerais do PEGASUS-32x construído na época eram:

1. Suporte para memória virtual.
2. Gerência de memória para relocação e proteção de programas em ambientes multiusuários e multitarefas.
3. UCPs da família MC680x0, com “cache”, para controle do Sistema Operacional e programas de usuário.
4. Instruções de 8, 16 ou 32 bits.
5. Barramento VME bus, com caminhos de 32 bits para dados e 24 bits para endereço.
6. Taxa de transferência de dados no barramento de 6.7 MBps.
7. “Cache” de 4 KB por UCP.
8. Velocidade de 1.2 Mips (1 UCP) a 3 Mips (4 UCPs).
9. Unidades de processamento perféricos (UPPs) inteligentes, utilizando o microprocessador Z80-A.
10. Placas auto-testáveis.
11. Rápida reconfiguração em caso de falhas, devido à existência de diversos processadores e à modularidade do projeto.

Após estudar diversos microprocessadores, concluímos que a linha MC680x0 da Motorola era a mais adequada, pois:

1. Apresentava arquitetura de 32 bits.

2. Apresentava instruções, modos de endereçamento, tratamento de exceções compatíveis com os atuais superminis.
3. Possuía alta velocidade de operação.
4. Permitia evolução para microprocessadores mais potentes.
5. Apresentava dispositivos auxiliares disponíveis (MMU, FPP, etc ...).

O Barramento do Sistema escolhido foi o VME bus, pois além de ser de 32 bits e não proprietário, é suportado pela Motorola e por uma série de companhias européias. Além disto, apresenta protocolo assíncrono (permite maior velocidade) e diversos níveis de interrupção e de controle de acesso ao barramento.

Por questões de modularidade, confiabilidade e expandibilidade, não existe um contrôle único para o sistema de memória, mas um contrôle distribuído, onde cada placa pode evoluir de 256 KB até 1 MB, quando são utilizadas pastilhas de 64 Kb. Com as novas pastilhas de 256 Kb, uma unidade de memória poderá conter 4 MB.

As Unidades de Processamento Periféricos tinham como objetivo mais importante, o desempenho do computador. Por isto procurou-se aliviar a carga dos processadores centrais em tarefas de entrada/saída, tais como discos, impressoras e terminais. Isto foi feito colocando-se um processador especializado em cada um dos módulos, o que contribuiu também para facilitar o auto-teste.

As Unidades Centrais de Processamento deveriam ter gerência de memória e “cache”para serem compatíveis em desempenho e potencialidade com as UCPs dos superminis. A gerência de memória do PEGASUS 32-x foi construída com integrados da linha TTL-Schottky, pois caso utilizássemos a MMU-68451 da Motorola, o sistema ficaria mais lento. Também foi construído um sistema de memória virtual para poder executar programas maiores do que a memória física.

O “Cache” fez-se necessário por 2 motivos:

1. Os microprocessadores são muito mais rápidos do que o sistema de memória, logo foi necessário haver uma memória muito rápida, interna às UCPs.
2. Não seria possível uma sistema multiprocessador sem “cache” nas UCPs, devido ao congestionamento do barramento do sistema.

Em um sistema com diversas UCPs, executando muitos programas, a solitação de disco é muito grande e normalmente torna-se o gargalo do sistema. Para resolver este problema, as unidades de interface de disco possuem um “cache” de blocos de disco, com capacidade de até 256 KB.

Para fazer com que o dados dos “caches” e da memória principal fossem sempre consistentes, construiu-se um mecanismo de vigilância das transferências no barramento. Se alguma unidade do sistema alterar o conteúdo da memória principal e este conteúdo residir também em algum “cache”, as posições correspondentes dos “caches” serão invalidadas. Isto possibilita implementar multiprocessamento e torna desnecessária a invalidação de todo o “cache” em transferências de acesso direto à memória.

Os sistemas da família PEGASUS-32x são formados a partir de múltiplos módulos escolhidos de cada um dos 7 tipos básicos, que são:

1. Módulo UCP com saída para duas consoles.
2. Módulo Unidade de Memória (UM) de até 1 MB.
3. Módulo UPP (Unidade de Processamento Periférico) para 16 terminais e 2 impressoras (UPPTI).
4. Módulo UPP para barramento “SASI” (Shugart Associates System Interface), que permite conectar discos “Winchester”, disco flexíveis e fitas do tipo “Streaming”.
5. Módulo UPP para 4 discos “SMD” (Storage Module Device) de alta capacidade.
6. Módulo UPP para fitas magnéticas.
7. Módulo UPP para interfaces especiais, como: “timer”, calendário, Processador de Ponto Flutuante, etc ...

Deve-se observar que uma configuração simples possuia apenas 4 módulos, ou seja, um módulo UCP com console, um módulo de memória com até 1 MB, um módulo UPP para até 16 terminais e duas impressoras e um módulo UPP para barramento “SASI”. Devido à alta homogeneidade do sistema, a evolução da família para a faixa de desempenho dos superminis se dá pela replicação dos módulos básicos.

O Sistema utiliza-se como meio de comunicação entre os diversos módulos o barramento “VME bus”, que é um padrão internacional. A mesma filosofia de utilizar padrões tipo “VME bus” e “SASI bus” foi seguida nos controladores de disco de alta capacidade, onde se utiliza o barramento “SMD”.

O Sistema possui ferramentas de depuração/teste, pois na fase inicial do desenvolvimento de um sistema baseado em microprocessador é de extrema importância a existência de um conjunto de ferramentas de diagnósticos e de depuração baseados em “software” à disposição da equipe de projetistas do “hardware” do sistema. O esquema de diagnosticar e depurar o “hardware” do sistema adotando o “software” como seu instrumento principal apresenta um série de vantagens, sendo a mais importante a redução do tempo de depuração.

Tendo em visto não tornar o desenvolvimento das ferramentas de diagnósticos e de depuração parte do caminho crítico do projeto, definiu-se um programa de diagnósticos e um depurador contendo os elementos essenciais para auxiliar a fase inicial da depuração da UCP do PEGASUS-32x.

Desde o início do projeto, foi de importância fundamental a interação entre as equipes de “hardware” e de “software” para realizar uma tarefa desta envergadura. Desta interação, podemos citar alguns itens, tais como:

1. Escolha das famílias de microprocessadores a utilizar.
2. Arquitetura do sistema.
3. Análise do conjunto de instruções para suportar as características do PLURIX.
4. Tratamento de erros no sistema.
5. Definição do sistema de gerenciamento da memória.
6. Definição dos acionadores de entrada/saída

As principais dificuldades foram, e continuam sendo, como geralmente ocorre nas universidades, de ordem material e financeira. No entanto, foi possível implementar, em tempo muito curto, um primeiro protótipo e provar que é possível construir no Brasil supermicrocomputadores muito potentes, dispensando qualquer importação de tecnologia.

As perspectivas para o PEGASUS 32-x são no sentido de se obter um desempenho maior ainda, desenvolvendo módulos que permitam, entre outras coisas:

1. Interligação de diversos PEGASUS-32x em um rede de alta velocidade (sistemas “loosely-coupled”).
2. Interligação de diversos PEGASUS-32x pela expansão do barramento paralelo de cada um (sistemas “tightly-coupled).

O grupo de PEGASUS-32x assim obtido é um sistema distribuído e tem desempenho similar a máquinas de grande porte. O sistema operacional distribuído (PLURIX-D) para suportar a nova arquitetura já estava sendo especificado.

O PEGASUS-32x ao centro, com unidade de fita magnética à esquerda, discos rígidos (Winchester) à direita e terminais de vídeo ao fundo.

O sucesso na construção do supermicrocomputador PEGASUS-32x e na implementação do Sistema Operacional PLURIX através das técnicas descritas provam a viabilidade da obtenção de sistemas computacionais de grande porte em países do Terceiro Mundo, sem a necessidade da importação da tecnologia do projeto. Obviamente a dependência tecnológica na área de componentes ainda é apreciável. Isto, entretanto, parece ser um fator menos crítico, já que existem inúmeros fabricantes internacionais que produzem circuitos similares, não havendo, portanto, necessariamente a dependência a um fabricante ou a um país específico.

O aproveitamento pela sociedade como um todo da tecnologia desenvolvida nas Universidades se faz através das indústrias nacionais. Seguindo a filosofia do NCE/UFRJ de repassar para a indústria os protótipos construídos na Universidade, esperava-se na época no mercado brasileiro a disponibilidade de sistemas PEGASUS-32x/PLURIX.

4. A EBC e o ÍCARUS

O PLURIX foi desenvolvido tendo-se em mente a sua utilização em computadores de 32 Bits. Nesta categoria encontram-se os supercomputadores, os "mainframes", os superminis e os supermicros. Embora os supercomputadores, os "mainframes" e os superminis não estarem fora do alcance de um transporte do PLURIX, o alvo principal a curto prazo eram os supermicros baseados nos microprocessadores Intel 80386, National 320x2, Motorola 680x0 e Zilog Z80000.

Como o PEGASUS-32x, o computador base para o qual o PLURIX foi inicialmente desenvolvido, utilizava o microprocessador 680x0 da Motorola, o transporte do PLURIX para sistemas utilizando microprocessadores também da linha Motorola ficava bastante simplificado.

Este era o caso do computador ÍCARUS (EBC-32010) produzido pela EBC (Empresa Brasileira de Computadores e Sistemas S. A.) do Rio de Janeiro, para o qual o PLURIX foi transportado. Tratava-se de um "desktop single-board computer" baseado no microprocessaodor 68010 da Motorola, muito semelhante aos que vinham sendo utilizados nos E. U. A. na época. Possuía de 1 a 8 MB de memória central , 8 terminais de vídeo alfa-numéricos e os clássicos discos "Winchester" (SASI), disquetes de 5 polegadas e fita "streamer". Já que o nome PLURIX (marca registrada do NCE/UFRJ) está reservado para identificar o desenvolvimento do sistema na Universidade, o transporte do PLURIX que controla o ÍCARUS foi conhecido comercialmente com É-IX.

5. A Versão 2.0 do PLURIX

A Versão 2.0 do PLURIX foi liberada pela UFRJ no dia 17 de agosto de 1988. Esta Versão sucedeu as Versões 1.0 e 1.1 liberadas em março e novembro de 1987, respectivamente. Com relação às versões anteriores, a Versão 2.0 do PLURIX tinha como característica principal o fato de ser um SOFIX (Sistema Operacional de filosofia UNIX) completo ("self-contained") não necessitando de nenhuma ferramenta externa ao PLURIX para permitir a sua completa compilação, distribuição, realização de cópias de segurança e utilização. As versões anteriores não incorporavam um compilador e exigiam a utilização de um compilador "C" para um microprocessador Motorola 680x0 que não pertencia ao PLURIX.

Utilizando-se apenas os diversos módulos de "software" que compõem a Versão 2.0 do PLURIX, sem utilizar nenhum dos inúmeros programas adicionais que foram desenvolvidos e adaptados por casas de "software", foi possível criar um ambiente computacional bastante completo. Desta forma, foi possível desenvolver-se "scripts" de "shell" sofisticados, já que o novo "shell" da Versão 2.0

incorporava programação, segundo o “Bourne shell”, e extensões similares ao “Korn shell”, ambos da AT&T. Era possível desenvolver-se “software” em linguagem “C” utilizando-se o compilador “C” do PLURIX, que faz parte da seqüência: pré-processador, compilador, montador para a família de microprocessadores Motorola 680x0 e editor de elos (“link-editor”).

Os programas podiam ser depurados em tempo real utilizando-se o depurador simbólico “wsdb”, que também faz parte da Versão 2.0. Com este depurador, que interage com o usuário através de múltiplas janelas, um programa pode ser executado em um ambiente controlado de forma que tanto o fluxo de execução de sua lógica como os valores de suas diversas variáveis reconhecidas diretamente pelos seus nome simbólicos podem ser acompanhados com muita facilidade.

A coordenação da atualização dos diversos módulos componentes de um sistema de “software” é feita pelo utilitário “make”, que na Versão 2.0 aproveita características de paralelismo do “hardware” quando ele existir. O novo “make” é capaz de importar variáveis do ambiente do programa de forma similar ao “shell”, e assim, facilitar em muito a tarefa de desenvolvimento de sistemas de “software” muito complexos.

Para a edição dos programas e de texto em geral é fornecido na Versão 2.0 um novo editor de tela “vi” integrado a um novo editor de linha “ex”. O texto preparado por esses novos editores ou mesmo pelos antigos pode ser formatado pelo novo “sroff”, que incorpora a possibilidade da criação automática de tabelas utilizando-se uma sintaxe semelhante à dos “tbl” do UNIX da AT&T.

O desenvolvimento de “software” aplicativo em geral, especialmente aqueles onde a interação com o usuário é muito importante, pode se valer das facilidades do pacote de rotinas “P-curses” que permite, de forma trivial, a criação de ambientes com janelas superpostas muito semelhantes aos que encontrávamos na época nos ambientes criados com PC’s.

Para o desenvolvimento e execução de “software” científico, o PLURIX Versão 2.0 oferece um conjunto completo de rotinas matemáticas de ponto flutuante que emulavam o co-processador de ponto flutuante da Motorola quando este não estava presente no sistema. Rotinas para cálculo científico como seno, logaritmo, etc. também estavam presentes na Versão 2.0, permitindo cálculos precisos com números de até 17 dígitos decimais.

No núcleo da Versão 2.0 do sistema PLURIX foram introduzidas as novas chamadas ao sistema “ptrace”, que permitem a implementação do depurador “wsdb”, e da “lockf”, que permite a utilização exclusiva de partes de arquivos, essencial para a construção e adaptação de “softwares” que gerenciam bancos de dados.

Finalmente, em atencão às empresas que já licenciaram o PLURIX e utilizando-se sugestões das mesmas, estabaleceu-se que as novas versões do PLURIX (incluindo a 2.0) sempre manteriam a compatibilidade com a versão anterior (neste caso a 1.1). Desta forma, todos os programas objeto que na época rodavam no PLURIX Versão 1.1 rodariam também sem problemas na Versão 2.0. Entretanto, somente os programas já desenvolvidos e compilados na Versão 2.0 podiam fazer uso, em sua plenitude, de todas as características desta última versão. A compatibilidade com versões mais antigas não era garantida. De qualquer forma, as empresas que licenciaram o PLURIX foram instruídas sobre as eventuais fontes de incompatibilidade quando da liberação da uma nova versão ou modificação.

As novas características da Versão 2.0 são descritas com mais detalhe abaixo e estão divididas nas seguintes partes:

1. Comandos novos: Novos comandos especialmente criados para Versão 2.0 ou antigos comandos de versões anteriores que foram totalmente reescritos. São eles: “cc”, “cmpobj”, “comm”, “file”, “lpkill”, “mac+”, “make”, “nice”, “sh”, “sroff”, “subst”, “tr”, “unique”, “untex”, “vi”, “walk” e “wsdb”.
2. Comandos estendidos: Comandos antigos que tiveram a sua funcionalidade melhorada e/ou estendida. São eles: “as”, “chmod”, “cptree”, “cmptree”, “gar”, “ld”, “man” e “show”.
3. Rotinas das bibliotecas novas: Novas rotinas da biblioteca da linguagem “C”. São elas: “curses”, rotinas flutuantes e matemáticas e “strtod”.
4. Rotinas das bibliotecas estendidas: Rotinas antigas que tiveram o seu funcionamento melhorado, estendido ou corrigido. São elas: “ftw”, “errlst”, “printf”, “qsort”, “scanf” e “setvbuf”.
5. Chamadas ao sistema novos: Novos serviços fornecidos ao usuário pelo núcleo do sistema operacional. São eles: “ptrace”, “sigchild” e “lockf”.
6. Chamadas ao sistema estendidos: Alterações do núcleo do sistema no sentido de melhorar, estender ou corrigir o funcionamento de serviços que já existiam em versões anteriores. São eles: estrutura “termio.h”, rotina “itocore”, módulo “fio/mount.c” e o módulo “sysfork.c”, chamada ao sistema “wait”.

6. A Similaridade PLURIX-UNIX

No dia 19.10.88 o NCE/UFRJ apresentou perante o Grupo de Assessoramento para Exame de Similaridade (GES) da SEI (Secretaria Especial de Informática) as características do Sistema Operacional PLURIX. Essa apresentação é uma conseqüência do fato de a SID Informática S. A. ter solicitado o cadastramento na SEI do seu sistema operacional baseado no UNIX da AT&T. Segundo a Lei de “Software” em vigor na época, as instituições que julgavam possuir produtos similares àqueles cujo cadastrameto estava sendo requerido deviam solicitar um exame de similaridade. Daí a criação de um grupo de assessoramento à SEI e realização da apresentação acima mencionada.

O GES, na reunião de 19.10.88, estava composto por quatro membros civis representando a ASSESPRO, a SBC a SUCESU e a USP, e três membros militares representando a Aeronáutica, o Exército e a Marinha. A comissão de estudo formada pelo GES e pela Subsecretaria Industrial da SEI tinha, segundo a regulamentação da Lei de ”Software”, um prazo de 120 dias para apresentar a conclusão de seu trabalho.

Além do NCE/UFRJ, também a COBRA foi chamada a apresentar o seu Sistema Operacional SOX e a LABO relatou a sua experiência no transporte do SOX. Após as apresentações, houve discussões entre os representantes das empresas convidadas (incluindo a SID), e os membros da comissão no sentido de prover maiores esclarecimentos sobre os sistemas apresentados.

Parte da equipe do PEGASUS-PLURIX-TROPIX: Manuel Lois Anido, Norival Ribeiro Figueira, José Luiz Ribeiro Filho, Newton Faller, Luiz Fernando Huet de Bacellar e Pedro Salenbauch

Embora o PLURIX não tivesse sido desenvolvido para ser igual ao UNIX da AT&T, ele nos parecia ser uma alternativa atraente. Contando com o apoio das empresas que licenciaram o PLURIX e já o estavam em vias de comerciá-lo (EBC do Rio de Janeiro e SISCO de São Paulo, respectivamente), o NCE/UFRJ julgou importante submeter o PLURIX a um exame de similiaridade por uma comissão independente.

No caso da aprovação da importação do UNIX da AT&T, as empresas nacionais se verão compelidas a licenciá-lo, por razões de mercado. Isto não impedirá, entretanto, que empresas nacionais continuem a promover os seus sistemas derivados do PLURIX, já que eles são tecnicamente tão bons ou melhores do que os derivados do UNIX da AT&T. A SISCO, declarou que no caso de o UNIX se tornar disponível no mercado brasileiro ela também vai tê-lo como opção em seus produtos. Entretanto, a primeira opção será sempre o seu sistema derivado do PLURIX. Isto se deveu ao fato de o mesmo ser tão bom quanto o importado, ser mais barato, não depender de contratos com fornecedores estrangeiros e a SISCO possuir total domínio tecnológico sobre o mesmo. Reconhece, entretanto, que a entrada do UNIX no Brasil, quando o derivado do PLURIX estiver um pouco mais consolidado no mercado (daqui a um ano, por exemplo) poderia em muito aumentar as chances do produto nacional.

Gostaríamos que os leitores estivessem informados sobre alguns dos documentos que foram solicitados pela comissão e enviados pelo NCE/UFRJ e que serviram de base para a análise. Em primeiro lugar, quanto ao desempenho do PLURIX, foi enviado o artigo “Avaliação de um ambiente UNIX com múltiplos processadores” do pesquisador do NCE/UFRJ Gabriel P. Silva, que foi publicado no Anais do Segundo Seminário Brasileiro de Arquitetura de Computadores e Processamento Paralelo em setembro de 1988. Neste artigo, mostra-se que, para uma série de “benchmarks” clássicos publicados na literatura internacional, quando executados em computadores com arquiteturas semelhantes, o desempenho do PLURIX é tão bom ou melhor do que o do UNIX.

O segundo documento enviado compara os comandos, utilitários, chamadas ao sistema e rotinas da biblioteca “C” com seus similares do UNIX tendo como base a segunda edição do “System V Interface Definition”. Apresentamos abaixo o índice da documentação enviado à Secretaria Especial de Informática, discriminando em quatro colunas as seguintes informações:

1. Nome do Programa ou Rotina do PLURIX;
2. Página do Manual do PLURIX onde encontrá-lo;
3. Nome do Programa ou rotina similar no UNIX;
4. Página do “System V Interface Definition” onde encontrá-lo.

A maioria dos nossos leitores certamente não possuem o “System V Interface Definition”. Entretanto, como interessados em UNIX, possivelmente possuem algum livro sobre o mesmo e/ou já o conhecem através da utilização de sistemas similares. Indepentemente do resultado da comissão da Secretaria Especial de Informática, poderão avaliar por si mesmo a similaridade entre os sistemas e julgar o que poderia ser feito com a Versão 2.0 do PLURIX.

Enquanto isto, empresas nacionais estavam viabilizando no mercado mais um produto nacional através da comercialização dos sistemas derivados do PLURIX, Versão 2.0 e nós, no NCE/UFRJ estávamos trabalhando com afinco na elaboração do PLURIX, Versão 3.0 que ficou disponível no segundo semestre de 1989.

7. O Sincronismo no Sistema Operacional PLURIX

Em um Sistema Operacional sofisticado há, em geral, diversos programas sendo processados simultaneamente. Uns estão realmente sendo executados por algum processador enquanto outros aguardam a sua vez para que a sua execução continue. Um programa em processamento em um determinado instante se consitui no que se denomina “processo”.

Um processo executa as instruções do programa do usuário até o ponto onde este chama o Sistema Operacional para pedir algum serviço (alocação de recursos, operação de entrada/saída, etc...). A partir deste ponto, as instruções do Sistema Operacional são executadas pelo processo até que o Sistema Operacional devolva o controle às instruções do usuário. Este processamento é considerado um único processo, parte em modo usuário e parte em modo supervisor (Sistema Operacional). Enquanto executando em modo supervisor, o Sistema Operacional pode decidir, por

alguma razão, interromper este processo e retornar à execução de algum outro. Esta operação tem o nome de “troca de contexto”.

A parte de um processo em modo supervisor, isto é, o programa Sistema Operacional, embora não funcione de forma diferente do programa de usuário, tem algumas características peculiares. A característica mais importante em relação à necessidade de sincronização deve-se ao fato de que, para a manutenção da consistência do sistema, determinadas seqüências de operações (regiões críticas) devem ser executadas de forma exclusiva. Isto é, enquanto um processo executa instruções que manipulem um recurso, todos os outros processos ficam proibidos de executar instruções que também iriam manipular o mesmo recurso.

Dijsktra, analisando os problemas de programação concorrente, definiu um mecanismo capaz de garantir a exclusividade de execução em ocasiões de manipulação de estruturas de dados. Este mecanismo foi chamado de “semáforo”, e na realidade tem funções mais amplas.

O semáforo, em sua forma mais geral, consiste de um número inteiro (valor do semáforo) sobre o qual são definidas duas operações fundamentais: “P” e “V”. A operação P é executada imediatamente antes de um trecho de programa onde a exclusividade é requerida enquanto V é executada imediatamente após. A operação P decrementa o valor do semáforo e se o resultado for positivo ou nulo, o processamento continua. Se ele for negativo, o processamento é suspenso até que o semáforo fique positivo. A operação V simplesmente incrementa o valor do semáforo.

O uso de semáforos em Sistemas Operacionais é feito através da associação do valor do semáforo ao número de recursos de um mesmo tipo disponíveis em um determinado instante. Quando o valor de um semáforo torna-se negativo, isto é, não há mais recursos disponíveis no momento, o processo que tentou, sem sucesso, alocar o recurso deve ter a sua execução suspensa, enquanto aguarda a liberação do mesmo. O processo pode aguardar pela liberação do recurso de duas maneiras distintas:

1. Desviando o processador para executar um outro processo que não está aguardando por nenhum recurso;
2. Mantendo o processador com o processo corrente testando continuamente a liberação do recurso (em malha).

A escolha entre as duas maneiras de se aguardar pelo recurso depende da relação entre o tempo previsto para a liberação do recurso e o tempo necessário para desviar o processador para executar um outro processo (troca de contexto). Se o tempo previsto para a liberação do recurso é muito menor do que o da troca de contexto, a opção 2 deve ser preferida. Caso contrário usa-se a opção 1.

Como regra prática, pode-se dizer que a espera de recursos que dependem de acesso a periféricos (blocos de discos, etc ...) deve ser feita através da opção 1, enquanto a espera de recurso que só dependem de acesso à memória (manipulação de listas, etc ...) deve dar preferência à opção 2.

O SPINLOCK é o semáforo básico do PLURIX. Ele implementa um semáforo de recurso único (exclusão mútua) e aguarda a liberação do recurso sem troca de contexto. O seu par, SPINFREE, libera o recurso.

O SLEEPLOCK implementa um semáforo de recurso único (também exclusão mútua, como o SPINLOCK), mas efetua uma troca de contexto caso o recurso não esteja disponível. O seu par, SLEEPFREE, libera o recurso.

O SEMALOCK implementa a forma generalizada do semáforo de Dijskstra. A cada chamada do SEMALOCK, o valor do semáforo é decrementado de uma unidade. Se o novo valor do semáforo for negativo, significa que todos os recursos já estão alocados e, como no SLEEPLOCK, uma troca de contexto é efetuada. O seu par, SEMAFREE, incrementa o valor do semáforo de uma unidade, indicando que um recurso foi liberado. O SEMALOCK é utilizado no PLURIX quando existem vários recursos de uma mesma espécie e um deles precisa ser alocado.

Um processo em execução pode aguardar não somente a liberação de um recurso. Algumas vezes ele aguarda a ocorrência de um evento que não necessariamente está associado a um recurso. Por exemplo, um processo pode querer ficar inativo durante 3 segundos, ou deve aguardar até que uma operação de entrada/saída termine, etc ... O EVENTWAIT faz com que um processo aguarde a ocorrência de um determinado evento e força a troca de contexto se este evento ainda não houver ocorrido. O seu par, EVENTDONE, sinaliza que o evento ocorreu.

O UNIX System V licenciado pela AT&T e as suas versões mais antigas (System III, Version 7, Version 6, etc ...) sob o aspecto de sincronização de processos, são sistemas mais simples, monoprocessados, e não requerem o ferramental sofisticado descrito neste artigo. Existia, entretanto, uma versão multiprocessada do UNIX em funcionamento apenas internamente nos laboratórios da Bell utilizando dois processadores. Começavam a surgir no mercado adaptações do UNIX para sistemas com muitos processadores, como foi o caso do DYNIX da Sequent Computers.

Em relação à sincronização de processos do UNIX multiprocessado mencionado acima, verificava-se uma unificação de todas as primitivas em torno do conceito de semáforo, utilizando as funcões PSEMA (alocação), VSEMA (liberação) e CPSEMA (teste), de acordo com a figura abaixo:

PLURIX	UNIX (multiprocessado)
SPINLOCK	while (! CPSEMA) /* vazio */ ;
SLEEPLOCK	init (1); PSEMA
SEMAINIT (n); SEMALOCK	init (n); PSEMA
EVENTWAIT	init (0); PSEMA

Repare que a função de inicialização (SEMAINIT, “init”) do valor do semáforo indicando o número total de recursos disponíveis, no PLURIX somente é necessário com a função SEMALOCK.

8. Aplicações de Tempo Real e o PLURIX

Por aplicações ditas de “tempo real”, entende-se a grosso modo, aquelas cujos resultados são usados imediatamente e, em geral, sem a intervenção do ser humano. Desta forma, os sistemas de controle automático, tanto de uma aeronave quanto de um simples elevador residencial são sistemas de tempo real.

Muitas aplicações de tempo real são complexas e necessitam do uso de computadores. Além disto, muitas delas requerem um tempo de resposta extremamente curto. Tais aplicações são escritas em um computador de forma dedicada, isto é , elas controlam totalmente o computador, não havendo diferenciação definida no “software” entre as funções específicas do aplicativo e aquelas relativas ao Sistema Operacional. O projetista de aplicações de tempo real tem de se preocupar com o sistema como um todo e não somente com a aplicação em si.

O uso de Sistemas Operacionais de tempo real vem facilitar em muito o trabalho do projetista de aplicações de tempo real. Certamente as aplicações que requerem um menor tempo de resposta possível para um dado computador continuarão a ser desenvolvidas de forma dedicada. A grande maioria delas, porém, pode se valer das facilidades oferecidas por um Sistema Operacional de tempo real.

O I.E.E.E. (Institute of Electrical and Electronics Engineers) define um Sistema Operacional de tempo real como aquele que executa as suas funções e responde a eventos externos assíncronos dentro de um intervalo de tempo determinístico (previsível). O tempo de resposta, dependendo da aplicação, pode variar de microsegundos a segundos, mas tem de ter um máximo bem definido, independente do estado do sistema.

Existem diversos Sistemas Operacionais apropriados para aplicações de tempo real. Alguns são simples monitores de processos computacionais, sem as sofisticações encontradas nos Sistemas Operacionais de uso geral. Há outros, entretanto, sofisticados o suficiente, não só para serem usados com um propósito geral, como também para controlar concorrentemente aplicações de tempo real.

Existiram alguns SOFIXs que, segundo seus fabricantes, eram apropriados para aplicações de tempo real. Podemos citar o UNOS da Charles River, o IDRIS da Whitesmiths e o REGULOS da Alcyon como exemplos. É importante notar que todos estes sistemas eram similares ao UNIX, mas tiveram de ser desenvolvidos independentemente dele (NÃO são derivados do UNIX, e portanto seus fabricantes não pagavam “royalties” à AT&T) simplesmente porque o UNIX não se prestava para aplicações de tempo real.

O PLURIX, embora estruturalmente diferente do UNIX, também não funcionava adequadamente para aplicações de tempo real. Desde a sua concepção, o PLURIX foi desenvolvido como uma alternativa ao UNIX e não houve, na época, preocupação em prepará-lo também para ser usado em áreas onde o UNIX não funcionava apropriadamente. Ambos os sistemas, entretanto, poderiam ser reformulados para tempo real.

As reformulações que o PLURIX e o UNIX deveriam sofrer são de natureza diversa devido às diferenças estruturais dos sistemas. As diversas características que um Sistema Operacional deve ter para torná-lo apropriado para aplicações de tempo real, acrescidas de uma análise de como introduzi-las tanto no PLURIX como no UNIX seguem abaixo.

1. Capacidade de manusear interrupções e entradas/saídas assíncronas para responder a eventos externos assíncronos em um tempo máximo determinado. Ambos os sistemas tem de ser reformulados para atender à esta característica.
2. Escalação de processos baseados em prioridades que podem ser definidas pelo usuário. O UNIX precisa ser reformulado. O PLURIX, que implementa um esquema de prioridades diferente do UNIX, pode ser facilmente adaptado para suportar esta característica.
3. Escalação de processos, com “pre-emption”, isto é, quando um processo de maior prioridade se torna disponível para ser executado, a execução do processo corrente, de menor prioridade, é interrompida imediatamente. O UNIX precisa de ser bastante reestruturado para atender esta característica. O PLURIX, embora necessite de adaptação, é baseado em um conjunto de primitivas de sincronização bem mais sofisticada do que o UNIX (veja o capítulo anterior). Isto certamente facilita em muito uma eventual adaptação.
4. Possibilidade de tornar um programa residente na memória, de forma a eliminar os acessos a disco que consomem muito tempo. O UNIX precisa de uma adaptacão. No PLURIX esta característica já é padrão.
5. Armazenamento de dados em arquivos fisicamente contíguos em disco ou alguma outra forma de minimizar o tempo de acesso a estes dados no disco. Neste aspecto ambos os sistemas requerem reformulação.
6. Facilidades de sincronização confiáveis para sincronizar e coordenar a execução de processos independentes e utilizar seus dados de forma compartilhada. O System V já tem muitas destas características enquanto que no PLURIX elas ainda não existem.
7. Desempenho tolerante a falhas, pois um sistema de tempo real não pode parar. O UNIX precisa ser reformulado. O PLURIX, como foi concebido para um ambiente de multiprocessamento, é naturalmente tolerante a falhas. Quando um dos dois processadores do PEGASUS falha, por exemplo, o outro continua controlando todo o sistema sem sequer se aperceber que está sozinho.

Ambos os sistemas, portanto, poderiam ser reformulados para atender os requisitos exigidos por aplicações de tempo real. A forma padronizada de como os SOFIXs de tempo real deverão apresentar essas novas características aos seus usuários foi objeto de estudo de uma subcomissão do POSIX, que foi definida em 1989.

No Brasil, os projetistas do PLURIX estudaram durante cerca de um ano as possibilidades de implementar um novo Sistema Operacional de tempo real baseado na experiência adquirida no desevolvimento do PLURIX.

9. A INFOCON e o PLURIX

A INFOCON, uma das empresas pioneiras em UNIX no Brasil, transportou os seus produtos de "software" para o ambiente dos SOFIXs, e assim, também para o PLURIX. O PLURIX aumentou assim sua biblioteca de "software" de apoio com produtos para processamento de textos, entrada de dados, controle de impressão em aplicações comerciais e conexão PC a supermicros PLURIX.

Dois dos produtos estavam disponíveis também para ambiente MS-DOS, permitindo a integração perfeita das atividades de seus usuários em MS-DOS e no PLURIX. Foi importante que existissem produtos com a mesma funcionalidade no MS-DOS e no PLURIX, principalmente para usuários que queriam racionalizar em treinamento de pessoal. Este era o caso típico de empresas com PCs e que então adquiriam supermicros rodando PLURIX.

O processador de textos INFOWORD e o sistema de entrada de dados LTD/INFOCON eram totalmente compatíveis com os "best-sellers" WORDSTAR da MicroPro e LTD da Cobra, e além disto, rodavam identicamente no PLURIX e no MS-DOS, o que era uma solução econômica e eficiente.

O INFOWORD era um processador de textos multilingue, que executava nos sistemas PLURIX e MS-DOS de maneira idêntica. Era totalmente compatível com o WORDSTAR, aproveitando todos os arquivos já formatados com este último processador. Usuários já familiarizados com o WORDSTAR no MS-DOS ou CP/M, podiam usar o INFOWORD no PLURIX ou no MS-DOS imediatamente. O INFOWORD dispunha ainda de um módulo de mala direta INFOMAIL, totalmente compatível com o MailMerge da MicroPro.

O LTD/INFOCOM era um sistema para transcrição e crítica de dados, totalmente compatível com o LTD da Cobra, para ambientes PLURIX e MS-DOS. O sistema era acompanhado de manuais didáticos e voltados para cada tipo de usuário: digitador, programador e administrador de sistema. O LTD/INFOCOM aproveitava todo o acervo de programas e arquivos de dados preparados com o LTD da Cobra e vinha com mais de 40 funções de críticas já embutidas. O uso do LTD/INFOCOM com os demais utilitários do PLURIX possibilitava o desenvolvimento de programas para entradas de dados em apenas 5% do tempo de programas em "C" ou COBOL.

O SPOOLMASTER era um servidor de impressão para o PLURIX, que substituía, com vantagens, o utilitário nativo "lpr". O SPOOLMASTER oferecia 17 comandos para controle total sobre o processo de impressão. A documentação do SPOOLMASTER consistia do manual do usuário, manual do administrador e do manual de referência.

Dentre as principais facilidades e recursos do SPOOLMASTER, destacavam-se:

1. Spool para qualquer dispositivo (fita magnética, impressora matricial ou laser, plotter, telex);
2. Controle simultâneo de várias impressoras;
3. Impressão em formulário escolhido pelo usuário;
4. Informação de estado completo de todos pedidos de impressão e de todas as impressoras;
5. Estatísticas de impressão;
6. Recuperação automática após queda de energia;
7. Informação sobre o instante de conclusão do pedido.

O SPOOLMASTER colocava na máquina PLURIX serviços de impressão antes só disponíveis em computadores de grande porte.

Para a comunicação PC-máquina PLURIX, a INFOCON associou-se com a SCIENCIA no desenvolvimento de um pacote para emulação de terminais e transferência de múltiplos arquivos entre PCs e máquinas PLURIX. O produto desenvolvido, AGIX, era compatível com o pacote americano TERM, eleito "software" do ano em 1987 no EUA.

O AGIX emulava terminais PLURIX em PCs e transferia arquivos entre ambientes MS-DOS e PLURIX e fazia ligações de PCs a máquinas PLURIX via conexão direta ou através de Modems em linha telefônica ou canal transdata. O AGIX transformava o PC em um terminal PLURIX, permitindo acentuação, cedilha, caracteres semigráficos, todos os atributos de vídeo e a utilização de todos os "softwares" disponíveis para o PLURIX.

O AGIX englobava também um módulo de compressão de dados, que reduzia o custo de chamadas telefônicas (via modem) em mais de 50%. Permitia a transferência de múltiplos arquivos bidirecionalmente, com conversão de formato PLURIX para MS-DOS (ou vice-versa), deteção e recuperação automática de erros e fornecimento de protocolo de transferência. O AGIX era residente na memória, permitindo ao usuário PC mudar instantaneamente o ambiente de trabalho, do MS-DOS para PLURIX e vice-versa, com um simples comando.

Os produtos da INFOCOM eram todos nacionais, categoria "A" na SEI e eram comercializados pela EBC em suas máquinas ÍCARUS 32010 e 32020.

10. A Avaliação do Desempenho do PEGASUS/PLURIX

A avaliação do desempenho de qualquer sistema computacional é sempre um processo difícil. Isto se deve ao fato de haver um grande número de variáveis envolvidas: o processador utilizado, o tipo de gerência de memória, a organização dos arquivos em disco, a linguagem escolhida e o compilador utilizado são algumas delas. Poder-se-ia acreditar que o melhor tipo de avaliação é aquela que executa a aplicação final do usuário nos diversos sistemas em avaliação. No entanto, esta avaliação, além de ser restrita, pode apresentar procedimentos incorretos, que favoreçam um sistema em detrimento de outro.

A avaliação realizada no PEGASUS/PLURIX procurou determinar o desempenho global do sistema, da maneira como se apresentaria ao usuário. Para isto, foi escolhida uma série de programas escritos em linguagem "C" encontrados na literatura especializada.

Como o PLURIX é um sistema similar ao UNIX da AT&T, a escolha de programas escritos em uma linguagem de alto nível permite, para efeito de comparação, a portabilidade destes programas para qualquer ambiente do tipo UNIX. O uso de linguagem simbólica ("assembly") se mostra inconveniente por estar distante das aplicacões do usuário, e ser dependente do processador utilizado.

O UNIX é um sistema operacional multiusuário e multitarefa. Desta forma, os diversos processos ocupam simultaneamente partes distintas da memória principal. No caso de haver pouca memória disponível, o disco poderá se tornar um recurso crítico, devido às trocas de contexto entre os diversos processos que competem pelo uso da memória e à excessiva utilização de arquivos temporários alocados denecessariamente em disco. Estas situações devem ser evitadas, quando possível, sob pena de o tempo de uso do disco mascarar completamente o tempo de uso dos demais recursos do sistema.

O PLURIX (no caso de haver bastante memória) permite que os programas de uso mais frequente fiquem permanentemente residentes na memória, juntamente com os arquivos temporários do sistema. Estas facilidades diminuem a utilização de disco, permitindo um melhor desempenho do sistema como um todo.

Antes de apresentarmos os programas de avalização utilizados, é conveniente esclarecermos alguns pontos que devem ser observados, para podermos realizar medidas não tendenciosas em um ambiente UNIX. Nas medidas realizadas, os programas foram executados em ambiente multiusuário para que o "overhead" normal do sistema pudesse ser levado em conta. Se utlilizássemos o modo monousuário (ou administrador), poderíamos obter resultados não confiáveis.

Para a medida do tempo de execução foi utilizada a rotina "time" do UNIX, que apesar de introduzir um pequeno "overhead", oferece um resultado bastante apurado. Esta chamada discrimina os tempos de usuário, supervisor e real, gastos pelo programa em execução (veja abaixo).

Cada programa foi executado mais de uma vez, e sem outros usuários no sistema. Os tempos mostrados são a média dos diversos resultados obtidos. Todo cuidado deve existir para o uso de opções de otimização e declaração de variáveis em registradores. Compiladores mais "espertos" podem modificar ou até mesmo suprimir trechos do programa original. Um caso típico é o compilador retirar uma malha que não altera o valor final de nenhuma variável.

Na comparação de desempenho entre vários computadores, é comum normalizarmos os resultados em relacão a um deles. Neste caso, deve-se utilizar a média geométrica do valor da comparação (razão entre tempos de execução), pois a utilização da média aritmética pode levar a uma interpretação errada dos resultados.

Para que o leitor possa avaliar corretamente os resultados apresentados, deve-se indicar a características do "hardware" da máquina avaliada: memória instalada, memória em disco, velocidade do processador e tipo do processador. Isto permite, por exemplo, explicar diferenças entre computadores do mesmo tipo, mas com configurações diferentes.

Os valores de saída apresentados pela função "time" de um sistema operacional tipo UNIX são os seguintes:

1. O tempo de usuário ("user") é o tempo gasto pelo processo executando instruções não privilegiadas;
2. O tempo de sistema ("sys") é o tempo gasto executando comandos privilegiados (por exemplo chamadas ao sistema) e mais algum "overhead" em nível de sistema operacional (a soma destes tempos costuma ser apresentada também como tempo de processador);

3. O tempo real (“real” ou “elapsed”) é o tempo decorrido entre o início e o término da execução do programa (não é a soma dos tempos de usuário e sistema. A diferença é o tempo perdido executando operações de entrada/saída e aguardando processamento nas filas do sistema).

Os pontos mais importantes para avaliação no UNIX são a interface com o usuário (“shell”), os dutos de comunicação entre processos (“pipe”), o acesso aos arquivos em disco, as chamadas ao sistema (“syscall”) e a capacidade de processamento de funções de usuário. Os programas escolhidos para avaliação do PEGASUS/PLURIX procuram exercitar estas funções, de modo que o desempenho obtido na execução destes programas estará perto daquele rendimento obtido por um usuário comum.

Os programas utilizados foram chamados de “syscall”, “funcall”, “piper”, “shell”, “sieve” e “disktest”, que executam os seguintes procedimentos:

1. O programa “syscall” realiza diversas chamadas ao sistema, para uma rotina que devolve o número de identificação do processo;
2. O programa “funcall” realiza diversas chamadas a uma subrotina de usuário;
3. O programa “piper” exercita a comunicação entre processos utilizando dutos;
4. O programa “shell” é um conjunto de diversos utilitários de uso comum ao UNIX;
5. O programa “sieve” calcula uma seqüência de números primos;
6. O programa “disktest” realiza uma série de acessos a arquivos em disco.

Estes mesmos programas também foram utilizados para avaliar o desempenho do PEGASUS/PLURIX operando com mais de um processador, com bons resultados.

Os resultados obtidos com a execução destes programas podem ser encontrados nas tabelas I e II, abaixo. Na tabela I é feita uma comparação com Supermicros, e na tabela II com Minis e Superminis. Os resultados mostram que o desempenho do PEGASUS/PLURIX é equivalente ao VAX 780, e superior aos demais computadores apresentados, entre eles um sistema desenvolvido pela AT&T.

O bom desempenho do PEGASUS/PLURIX pode ser atribuído à sua arquitetura, com controladores inteligentes de entrada/saída e a memória “cache”; e ao cuidado de projeto do sistema, que procurou alcançar uma solução integrada de “hardware” e “software”, adaptando um às necessidades do outro.

A avaliação realizada procurou mostrar as potencialidades do sistema PEGASUS/PLURIX e, considerando o fato de este não ser um produto comercial, mostrou excelentes resultados. O transporte do Sistema Operacional PLURIX para outras arquiteturas permitiria que este mesmo conjunto de programas fosse executado em computadores comerciais. Com isto poderíamos obter resultados da comparação com outros sistemas de ambientes UNIX que foram lançados no mercado.

Os resultados das avaliações certamente auxiliarão o usuário final na escolha de um sistema adequado às suas necessidades.

TABELA I

Nome do Programa	AT&T 3B2	ALTOS 586-30	ZILOG 11+	PEGASUS 32.1
Sieve	7.1 (7.0)	7.0 (6.9)	5.0 (4.9)	3.1 (3.0 + 0.0)
Syscall	3.5 (3.4)	5.5 (4.8)	4.0 (4.0)	1.5 (0.1 + 1.4)
Funcall	48.0 (47.8)	37.0 (36.8)	15.0 (14.9)	18.9 (18.8 + 0.1)
Piper	10.2 (5.3)	11.0 (6.5)	9.0 (3.9)	10.5 (0.1 + 4.6)
Disktest	30.3 (12.5)	70.0 (15.1)	30.0 (10.7)	33.8 (0.2 + 15.1)

TABELA II

Nome do Programa	VAX-780	VAX-750	PDP 11/70	PEGASUS 32.1
Sieve	1.7 (1.5 + 0.1)	2.4 (1.7 + 0.1)	2.3 (1.6 + 0.1)	1.6 (1.5 + 0.0)
Syscall	4.8 (0,4 + 4.0)	7.0 (0.8 + 6.2)	8.0 (0.2 + 7.5)	3.8 (0.3 + 3.5)
Shell	3.3 (0.3 + 1.8)	3.8 (0.4 + 1.5)	4.0 (0.2 + 1.7)	3.6 (0.3 + 2.8)
Piper	3.2 (0.1 + 1.2)	4.6 (0.2 + 2.1)	8.1 (0.0 + 3.4)	4.5 (0.0 + 2.0)

As características dos computadores dados nas tabelas são:

4. AT&T 3B2: 2 MB de memória, 32 MB de disco, UCP 32100, Unix Sistem V.
5. ALTOs 586-30: 512 KB de memória, 30 MB de disco, UCP 8086, Xenix Versão 7.
6. ZILOG 11+: 512 KB de memória, 52 MB de disco, UCP Z8000, Zeus Versão III.
7. PEGASUS: 4 MB de memória, 85 MB de disco, UCP 60020 @ 12 MHz, Plurix V 1.0.
8. VAX-780: 4 MB de memória, 256 MB de disco, Unix 4.1 BSD.
9. VAX-750: 2 MB de memória, 121 MB de disco, Unix 4.1 BSD.
10. PDP11/70: 1 MB de memória, 400 MB de disco, Zeus Versão III

Os tempos indicados nas tabelas são em segundos : Real (user + sys).

11. Redes de Comunicações para os SOFIXs

Os computadores têm permitido ultimamente uma revolução nas áreas industrial, comercial e de serviços devido à possibilidade da comunicação entre eles. Na produção industrial, máquinas automáticas são ligadas diretamente aos computadores de projeto e aos computadores da administração. No comércio, computadores controlam automaticamente os estoques, por estarem diretamente conectados aos computadores chamados ”ponto de venda”. Um cliente de um grande banco no Brasil, por exemplo, dificilmente poderia hoje entender como os serviços ali prestados

poderiam ser executados sem uma comunicação direta entre os diversos computadores da instituição bancária. Os conjuntos de computadores assim entreligados numa rede de comunicações têm efetivamente revolucionado a vida moderna.

Muitas redes comerciais existiam na época, no mundo. Embora algumas delas tivessem âmbito internacional, elas ainda eram restritas, muitas delas particulares, dando a impressão (correta) de fragmentação na comunicação entre os computadores. O ideal seria uma visão unificada onde a conexão entre computadores se fizesse de maneira semelhante às ligações telefônicas, onde o usuário, ao discar um simples número, pode acessar qualquer aparelho telefônico do mundo.

A ligação global de um sistema de comunicações entre computadores esbarra num dos mais complexos problemas a ser resolvido internacionalmente: padronização. Este problema não é somente técnico, mas envolve aspectos poliíticos e econômicos. Diversos fabricantes de computadores já tinham a sua rede instalada usando a sua própria definição não só da parte física (elétrica) como da parte lógica (protocolos). A parte física pode ser padronizada de uma forma mais fácil pelo fato de que estes problemas são similares aos já enfrentados há um bom tempo pelos sistemas de telecomunicações tradicionais (telefonia, telex, televisão, etc ...) existindo inclusive órgãos internacionais como o CCITT (Comité Consultatif International Téléphonique et Télégraphique) que já há muito tempo definem tais padrões. A parte de protocolos, entretanto, é mais difícil de ter um padrão aceito internacionalmente a curto prazo. Esforços, entretanto, existiram, e a proposta ISO/OSI (International Standards Organization / Open Systems Interconnection) é a mais importante.

Entendendo a importância da comunicação entre computadores independentemente do seu tipo de fabricante, aliada ao fato de que não existiam ainda (pelo menos na época) padrões estabelecidos para tais comunicações, o DARPA (Defense Advanced Research Projects Agency) do governo americano resolveu financiar a partir dos meados dos anos 1970 a definição de um novo conjunto de protocolos e suas respectivas implementações em diversos computadores. A experiência acumulada com a utilização da rede de pacotes ARPANET (Advanced Research Projects Agency Network) facilitou este empreendimento. Este conjunto de protocolos foi chamado de TCP/IP (Transmission Control Protocol / Internet Protocol) que são os nomes dos dois protocolos principais do conjunto.

A primeira conexão usando TCP/IP data de 1980 e já em 1983 havia sido completada a transição da ARPANET para a utilização do conjunto TCP/IP como protocolo padrão. Diversas redes locais (“networks”) com o incentivo do DARPA, passaram, assim, também a utilizar o TCP/IP e se conectar a outras redes locais através da ARPANET e posteriormente, através de outras redes de longa distância. As redes locais, assim conectadas, passaram a ter um papel ativo nessa rede global criando o que passou a ser chamado de “internetwork” ou simplesmente INTERNET.

A importância da INTERNET está no fato de que computadores de quaisquer fabricantes poderem comunicar-se utilizando apenas um endereço, que é similar a um número de telefone. O sistema é distribuído, pois, cada rede local administra o seu tráfego interno. Computadores que controlam a conexão entre duas redes são chamados de “gateways” porque desempenham o papel de porteiras para o exterior de uma rede. A definição dos endereços, entretanto, é centralizada para evitar duplicações.

A importância da INTERNET para os SOFIXs é que todo o desenvolvimento financiado pela DARPA foi feito utilizando-se e estendendo-se o sistema operacional UNIX desenvolvido na Universidade da California em Berkeley, E.U.A. Com a disseminação deste sistema, chamado BSD (Berkeley Software Distribution), após 7 anos de existência (1987) a INTERNET já conectava alguns milhares de redes locais no E.U.A. e Europa interligando mais de 20000 computadores em universidades e laboratórios de pesquisa. Desde então, o número de conexões à INTERNET tem crescido constantemente. Portanto, devido ao histórico deste desenvolvimento, os protocolos TCP/IP são atualmente não só um padrão de fato para os SOFIXs, como têm sido implementados em inúmeros outros Sistemas Operacionais.

Os serviços básicos oferecidos pelos SOFIXs que se utilizam do TPC/IP eram na época: correio eletrônico, transferência de arquivos e acesso remoto. Com o correio eletrônico é possível enviar-se uma mensagem para qualquer usuário conectado à INTERNET; é possível transferir-se arquivos cujo acesso tenha sido autorizado entre quaisquer computadores conectados à INTERNET; e é possível utilizar-se remotamente os recursos computacionais (autorizados) de qualquer computador conectado à INTERNET. Embora estes serviços não sejam uma exclusividade da INTERNET, a forma como a INTERNET e seus protocolos foram definidos faz com que a comunicação seja extremamente eficiente e confiável. É eficiente porque os pacotes que compõe as mensagens podem seguir até o destino por caminhos diferentes (sempre escolhendo o caminho mais rápido, onde haja menos congestionamento). É confiável porque a confirmação do recebimento de cada pacote é feita pelo computador de destino diretamente ao computador de origem, e não por computadores intermediários que posteriormente poderiam perder os pacotes de dados no caminho.

Com relação à padronização internacional, a INTERNET está em contínua evolução e tem se orientado pelos seguintes princípios: usar um protocolo internacional, sempre que ele exista e se aplique ao problema a ser resolvido; inventar novos protocolos apenas quando os padrões existentes sejam insuficientes, mas estar sempre preparado para migrar para protocolos padrão internacionais assim que se tornem disponíveis. Desta forma já se prevê, por exemplo, nos próximos 5 anos a migração do protocolo TCP para o TP4 da ISO ([1]).

Como os protocolos TCP/IP são mais simples do que os da ISO e ainda terão uma longa vida, decidiu-se implementá-lo no PLURIX de forma a permitir a comunicação com outros computadores da INTERNET. A implementação destes protocolos no PLURIX abriu não só uma interessante área de pesquisa para a equipe de desenvolvimento do PLURIX como também permitiu a utilização de "software" de comunicação importado. Isto é possível, já que a interface que se preparou para a Versão 3.0 do PLURIX é similar à do UNIX de Berkeley, onde foi originalmente desenvolvido o TCP/IP.

12. Interfaces Gráficas para os SOFIXs

A interação de um usuário com um computador tem-se feito cada vez mais freqüêntemente através de uma interface gráfica. Esta interface gráfica apresentada na tela de um terminal ou da

[1] Nota do Editor: Esta previsão não se concretizou

própria console do computador tem-se caracterizado por figuras (ícones), que estão associadas à funções disponíveis para o usuário, e por espaços delimitados (janelas), onde textos e/ou imagens podem ser exibidos. O usuário interage com os ícones e as janelas através de um dispositivo posicionador ("mouse", em geral) e de um teclado alfanumérico e/ou funcional. Um dispositivo de impressão de imagens (impressora a "laser", em geral) e de entrada de imagens (rastreadores) também podem ser considerados como elementos componentes da interface gráfica.

A interface gráfica mais popular na época era a que encontrávamos no computador Macintosh da Apple. É impressionante como praticamente qualquer pessoa pode utilizar as funções de um Macintosh quase sem nenhum treinamento prévio. A sua forma de interação se tornou tão agradável ao usuário que praticamente todos os fornecedores de sistemas operacionais têm procurado imitá-la nos seus produtos. Por isto, a Apple vem tentando até patentear a aparência e funcionamento de sua interface gráfica.

Os SOFIXs, que sempre foram identificados como sistemas cuja interface com o usuário era "não amigável", também começavam a oferecer interfaces gráficas. E como uma interface gráfica somente passa a ser importante se houver um conjunto razoável de "software" desenvolvido para ela, havia uma disputa para se definir uma interface gráfica padrão para os SOFIXs.

Dentre as interfaces gráficas mais promissoras para se constituir em um padrão para os SOFIXs, podíamos citar a OPEN LOOK, da AT&T, defendida também pela SUN e pela UNIX International, naturalmente, e a MOTIF, da Open Software Foundation, defendida pela Digital. Não podemos esquecer também da PM/X da Microsoft, uma adaptação para o ambiente UNIX do Presentation Manager do OS/2, e o NEXT STEP, da NEXT, defendida também pela IBM. Qual delas iria ser padrão, se que é que alguma, somente o tempo diria. Importante, porém, é que as interfaces gráficas que almejam o "status" de padrão deveriam poder ser amplamente licenciadas a um custo relativamente baixo.

A implementação, ou mesmo a simples implantação, de uma interface gráfica em um SOFIX é, naturalmente, uma tarefa não trivial. Assim como no caso de comunicações, em que se definiram os célebres sete níveis para a implementação de protocolos, também com as interfaces gráficas tem se falado em níveis, embora sem uma formalização oficial. A título de analogia, vamos descrever como são divididas as funções nos sete níveis, arbitrariamente escolhidos. Devemos lembrar que em muitos sistemas reais, diversos níveis contíguos se confundem por terem sido implementados como um único conjunto de facilidades.

Os sete níveis são os seguintes:

1. O nível 1 é o nível do "hardware". Neste nível estão as funções básicas oferecidas pelo terminal ou console gráficos. Muitas vezes estes terminais possuem processadores gráficos que muito facilitam a implementação de funções de nível mais alto.
2. O nível 2 é o nível da comunicação física do terminal gráfico com o programa aplicativo. Se uma console (local) é utilizada, este nível é nulo.
3. O nível 3 é o da comunicação lógica de terminal/console gráficos com o programa aplicativo. Esta comunicação pode ser extermamente dependente dos níveis mais altos, tornando a sua utilização por fabricantes distintos, quase impossível. Já prevendo estes problemas, o MIT (Massachussets Institute of Technology) dos E.U.A. definira um protocolo padronizado

e o colocaram em domínio público. Isto vem fazendo com que não só neste nível, como também nos dois seguintes, os fabricantes venha se decidindo a usar a padronização do MIT através das definições dos “X-Window System”, ou “X-Windows” como vem sendo popularmente (embora erroneamente) chamado. Para este nível, o “X-Windows” define o “X-Protocol”.

4. No nível 4, encontramos as rotinas básicas que geram chamadas ao protocolo lógico. Aqui se faz a conexão com as linguagens de programação (“language binding”). Neste nível estão as funções que manuseaim janelas, desenham polígonos, pintam superfícies, etc. Aqui também a biblioteca “X-Lib” do “X-Windows” vem sendo considerada como um padrão, embora existam interfaces gráficas que se utilizam do “X-Protocol” mas não usam a “X-Lib”.
5. No nível 5, encontramos a interface com os programas aplicativos (Aplication Program Interfaces). Aqui encontramos conjuntos de rotinas para a construção de elementos básicos como cursores gráficos, molduras de janelas, etc. Como cada fabricante começou a inventar as suas próprias ferramentas neste nível, o MIT decidiu fornecer pelo menos um conjunto básico padronizado chamado “Xt Intrinsics”. Algumas interfaces gráficas, mesmo utilizando os níveis mais baixos do “X-Windows” não fazem uso do “Xt Intrinsics”.
6. No nível 6, encontramos as definições dos componentes da interface gráfica, isto é, os elementos que vão aparecer na tela gráfico. Aqui definem-se os tipos de “menu”, formas de botões e desenhos de ícones. Na terminologia do “X-Windows” são chamadas “widgets” e não são padronizados. Assim temos os “HP Widgets”, “Sony Widgets”, etc. Alguns destes “widgets” são patenteados e não podem ser utilizados sem uma licença dos seus proprietários.
7. Finalmente, no nível 7, encontramos a aparência e funcionamento da interface gráfica. Aqui define-se como os “widgets” interagem com outros “widgets” e com o usuário para produzir o efeito desejado pelo programa aplicativo. Este é o nível de definição onde se encontra o que normalmente é reconhecido como a interface gráfica propriamente dita. O “OPEN LOOK” por exemplo, enquadra-se neste nível, embora se utilize das definições do “X-Windows” em outros níveis.

Para a versão 3.0 do PLURIX, que iria ser liberada no segundo trimestre de 1990, estava sendo desenvolvido um “X-Terminal”, isto é um terminal gráfico de alta resolução que pode ser diretamente conectado a uma aplicação executada em um computador que utilize o “X-Protocol”. Inicialmente este terminal estava conectado ao supermicro PEGASUS-32x como o sistema operacional PLURIX. Também o PLURIX estava sendo estendido para suportar as novas funções gráficas.

Como a conexão de “X-Terminals”, em geral, tem sido feita no nível 2 através do protocolo TCP/IP com comunicação tipo “Ethernet”, as interfaces tipo “Ethernet”, assim como os protocolos TCP/IP e as extensões do PLURIX já vinham sendo implementadas de forma a permitir que o PLURIX versão 3.0 fosse um sistema distribuído.

13. X-Windows – Um Padrão de fato em Interfaces Gráficas

As interfaces gráficas através das quais um usuário pode interagir de forma confortável com um Sistema Computacional pode, para uma melhor compreensão, ter a sua forma de implementação dividida em níveis arbitrários. No capítulo anterior foram usados sete níveis, entretanto no artigo "A Guide to Graphic User Interfaces" (Byte, julho de 1989), o número de níveis escolhido foi cinco.

Para a implementação dos níveis 3 e 4, segundo o capítulo anterior (que correspondem ao nível "Windowing System" do artigo da Byte, das interfaces gráficas em praticamente todos os SOFIXs (exceto o sistema da NEXT) é utilizado um sistema de janelas de "X-Windows". Embora ainda não homologado como um padrão, o "X-Windows" é na realidade aceito por todos os implementadores de sistemas similiares so UNIX como o que tinha maiores possibilidades de se tornar realmente um padrão. Isto se deve ao fato de que uma grande quantidade de "software" já vinha sendo produzida utilizando o "X-Windows".

O "X-Windows" foi desenvolvido no MIT e a letra que indica o seu nome origina-se de uma seqüência de sistemas de janelas onde estão o VGTS e W, ambos desenvolvidos na Universidade de Stanford em cima do Sistema V, um "software" básico para sistemas distribuídos. A idéia inicial para o desenvolvimento do sistema partiu da necessidade de interligar as diversas estações de trabalho gráficas, de diferentes fabricantes, existentes no MIT, e esta ligação deveria ser estendida aos computadores de maior porte. Além disto, existia a intenção de gerar um ambiente baseado na filosofia de criação de janelas associadas a cada tarefa que o usuário estivesse executando.

Desta forma, o sistema "X-Windows" foi projetado para ser um sistema gráfico, com suporte para janelas, multitarefa, distribuído, capaz de acessar redes de comunicação heterogêneas de forma transparente e independentemente dos tipos de máquinas existentes na rede. A forma de se ter independência com relação ao tipo de máquina é obtida através da tarefa de desenhar em 2 partes, usando o modelo cliente/servidor. Através deste modelo, uma aplicação (cliente) que esteja executada em uma determinada máquina passa os comandos gráficos necessários para o servidor, que pode estar na mesma máquina ou em outra máquina distinta, e este então irá executar o que lhe foi pedido. O sistema "X-Windows" padroniza o protocolo de comunicação ("X-Protocol") entre o programa cliente e o servidor, de forma que qualquer servidor que esteja conectado ao sistema possa executar o que foi requisitado.

O protocolo de comunicação foi projetado para ser implementado em cima de qualquer canal de comunicação assíncrono de largura igual a 1 Byte e que tenha uma velocidade de transmissão aceitável para o tipo de aplicação desejada. O Sistema fornece uma biblioteca ("X-Lib"), que quando conectada à aplicação, fornece as rotinas necessárias para comunicação com o servidor através do protocolo definido. Do outro lado do canal de comunicação devem existir um ou mais servidores capazes de reconhecer este protocolo. Todos irão escutar os pedidos que passam no canal, porém só atenderão àqueles endereçados ao próprio servidor.

A biblioteca "X" fornece não somente as rotinas necessárias para comunicação com o servidor como também funções gráficas e rotinas que facilitam as operações com janelas. Para um programador no sistema, tanto a forma de comunicação com o servidor, como a maneira de executar as operações gráficas desejadas, são feitas de maneira totalmente transparentes quanto ao

tipo de máquina que irá exibir o resultado da aplicação, a localização desta máquina e o Sistema Operacional nela executado. Para este programador bastará conectar a sua aplicação à biblioteca “X-Lib” e especificar o nome ou endereço da máquina onde deseja exibir sua aplicação.

É no servidor do sistema (“X-Server”) que são encontradas e concentradas todas as dependências com relação ao dispositivo que irá exibir o resultado das aplicações. O servidor é um programa que tem a sua execução normalmente iniciada junto ao início da execução do Sistema Operacional do computador onde ele irá funcionar. Durante todo o período de funcionamento desta máquina, o servidor estará em operação recebendo os comandos vindos pelo canal de comunicação ao qual está conectado. Em cada máquina deve existir um servidor para cada dispositivo de exibição (“display”), sendo que cada um destes dispositivos normalmente contém uma ou mais telas (monitores gráficos), um teclado alfanumérico e um dispositivo de apontamento (normalmente um “mouse”). Uma mesma máquina pode ter vários servidores, um para cada dispositivo de exibição que possuir. O servidor irá realizar a ligação entre o canal de comunicação e o dispositivo de exibição, executando as requisições dos diversos clientes e exibindo-as nas suas telas quando for necessário, e recebendo os eventos ocorridos no teclado e no dispositivo de apontamente, passando para o cliente apropriado.

Apesar de ter sido desenvolvido desde o início para o Sistema Operacional UNIX (mais especificamente para o Sistema UNIX da Universidade da California, em Berkeley), e, pela facilidade de transporte, a isto se deve o fato de sua grande popularidade, o sistema “X-Windows” não está amarrado somente a este tipo de Sistema Operacional. Ele pode ser implantado sobre qualquer Sistema Operacional, o que, por exemplo, já foi realizado pela DEC para o “VAX-VMS”. Entretanto, o transporte do sistema para computadores com Sistemas Operacionais que não possuam facilidades para comunicação entre processos, suporte para comunicação atráves de rede de computadores e capacidade para executar múltiplas tarefas, é bem mais custosa do que para aqueles que possuam tais características.

As características descritas levaram o sistema “X-Windows” a ser um grande sucesso e, a partir da versão 10.4, o MIT tornou-se um sistema de domínio público, passando a distribuí-lo a um custo simbólico. Desde então, firmas como a DEC, Sun, HP, IBM, Apollo e outras se juntaram ao MIT, formando um consórcio (“X-Consortium”), com a finalidade de difundir o sistema, aperfeiçoá-lo, implantá-lo em seus computadores e estações de trabalho.

Em março de 1988 foi lançada uma nova versão do sistema, 11.2, também de domínio público. Esta versão contou com a colaboração de diversos pesquisadores das firmas que se juntaram ao MIT no desenvolvimento do sistema “X-Windows”. Por este motivo, a versão 11.2 é bem mais completa, documentada, diversos erros das versões anteriores foram corrigidos e, principalmente, foi divulgado aquilo que acreditava-se que fosse a versão final sobre a especificação do protocolo de comunicação entre o cliente e o servidor.

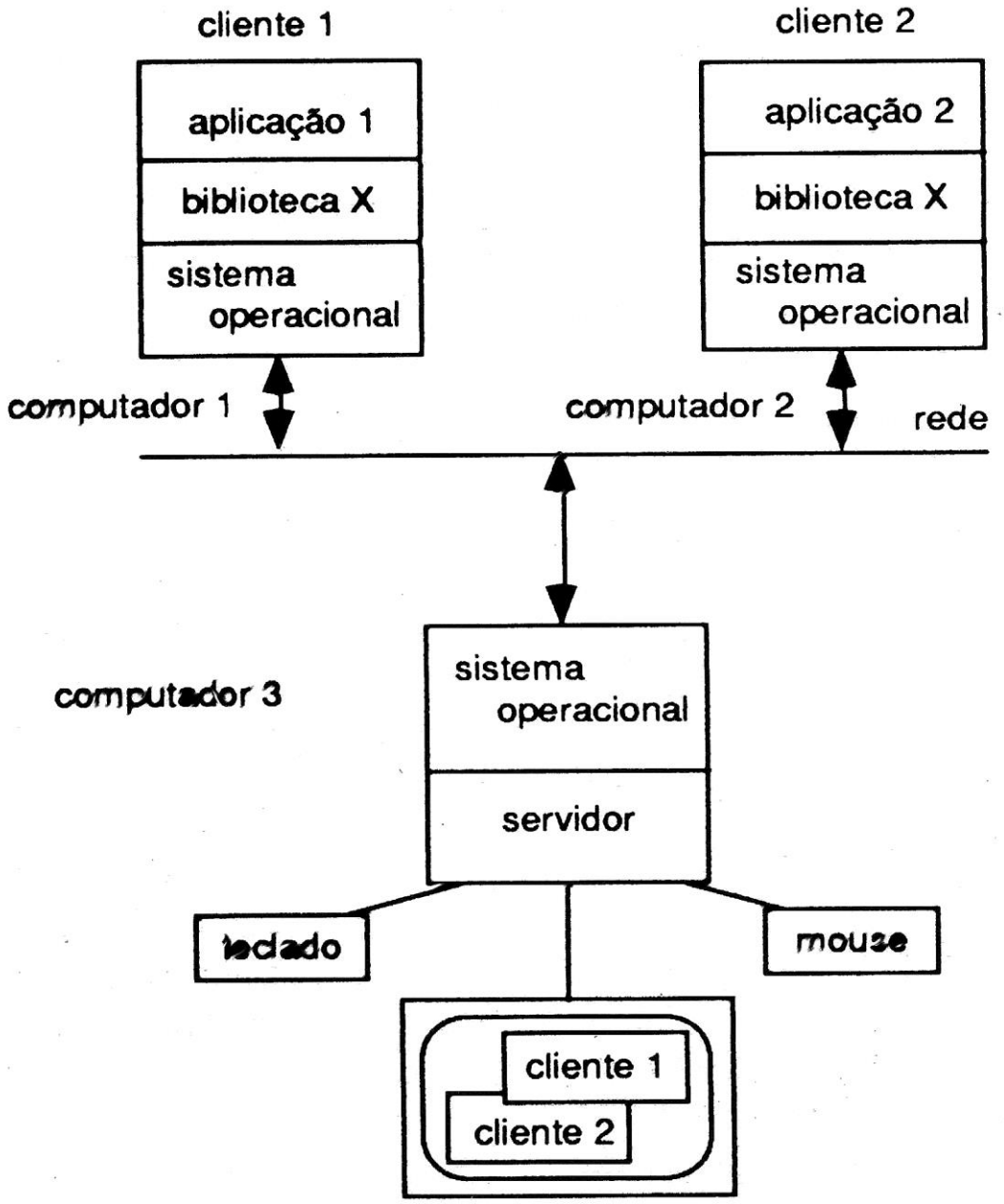
A partir de então, observa-se cada vez mais a difusão do sistema por diversos tipo de usuários assim como a adesão de outros fabricantes ao mesmo. Empresas como a AT&T, SUN, Sony, DEC, HP e mesmo a “Open Software Foundation”, com a “Motif” estão transportando ou desenvolvendo suas interfaces gráficas com o usuário, proprietárias para o sistema “X-Windows”. Esta fórmula, o usuário das mesmas não só passa a ter acesso a uma ambiente gráfico realmente distribuído, como também pode desenvolver programas aplicativos que utilizam o sistema “X-Windows” diretamente. Estes

programas podem ser facilmente transportados para outras máquinas, inclusive de fabricantes distintos, desde que estas também possuam a mesma versão do sistema “X-Windows”.

Com a grande difusão do sistema “X-Windows”, diversas firmas especializadas no desenvolvimento de “software” aplicativo passar a adotá-lo como padrão para o desenvolvimento de suas aplicações. Por este motivo, com excessão das aplicações que necessitam de alto desempenho, e para estas são utilizados outros sistemas específicos, o sistema “X-Windows” tem se tornado um padrão de fato para aplicações gráficas, pelo menos nos níveis mais baixos. Enquanto isto, aguardava-se que os grandes fabricantes internacionais chegassem a uma definição se teríamos ou não um padrão nos níveis mais altos das interfaces gráficas dos Sistemas Operacionais com filosofia UNIX.

Entendendo a importância de uma interface gráfica para interação com o usuário com um sistema computacional aliada à tendencia do “X-Windows” torna-se um padrão para os SOFIXs, o NCE/UFRJ deu início, em outubro de 1988 ao desenvolvimento de um projeto para integrar o “X-Windows” ao PLURIX. Este projeto deveria estar concluído em 1990.

Na figura abaixo, temos um modelo cliente/servidor do sistema “X-Windows” implementado através de uma rede de computadores. Na rede circulam mensagens definidas pelo protocolo de comunicação cliente/servidor do sistema “X-Windows”.



14. Implementação de uma Inferface Gráfica para o PLURIX

O advento das estações de trabalho, com suas telas de exibição de alta resolução, trouxe o potencial das interfaces gráficas para uma variedade de aplicações. As interfaces gráficas facilitam o processo de aprendizado, uso e entendimento das aplicacões. Diversas interfaces gráficas

proprietárias surgiram, provocando dificuldades no transporte de aplicações desenvolvidas em um sistema computacional, que utiliza uma terminada interface, para outro com interface distinta.

Diante deste fato, surgiu a necessidade da busca de uma interface gráfica que fosse padrão em sistemas computacionais de diferentes fabricantes. A partir desta idéia, foi desenvolvido no MIT o sistema de janelas "X-Windows" (veja o capítulo anterior). Este sistema tem sua arquitetura dividida em diversas camadas ou níveis. Essas camadas se sobrepõem conforme aumenta a complexidade das bibliotecas gráficas oferecidas ao usuário. A finalidade é a de permitir ao usuário a utilização das diversas bibliotecas, de acordo com as necessidades de sua aplicação.

O sistema "X-Windows" foi colocado em domínio público e a sua arquitetura passou a ser adotada com um padrão em sistemas gráficos. No entanto, somente a camado do nível básico do sistema "X-Windows" passou a ser utilizada como um padrão de fato. Foi iniciada uma imensa discussão sobre como seriam definidas as camadas de mais alto nível que utilizariam as rotinas do nível básico para fornecer ao usuário um ambiente gráfico sofisticado. Esta discussão se prolongou durante anos, até que se tenha chegado a um consenso sobre a padronização das camadas de mais alto nível.

Simultâneamente a estes acontecimentos, foi desenvolvido no NCE/UFRJ o Sistema Operacional PLURIX. O PLURIX é executado (entre outros) no computador PEGASUS-32x, também desenvolvido no mesmo local. Surgiu então a necessidade de adotar o PLURIX de uma interface que permitisse aos seus usuários o desenvolvimento de aplicações gráficas. Porém, o PEGASUS-32x não possuía o suporte de um "hardware" gráfico. Como o PEGASUS-32x foi projetado como computador protótipo em 1984, procurou-se uma solução que se tornasse independente deste computador.

A solução encontrada foi o desenvolvimento de um "X-Terminal". Estes sistemas computacionais têm a sua arquitetura baseada no sistema "X-Windows" e funcionam basicamente como servidores de exibição gráfica. Somente as camadas do sistema "X-Windows" que vão até o nível básico são necessárias para a implementação dos "X-Terminals". Desta forma, adotou-se uma solução gráfica para o PLURIX que possibilitaria a implantação das demais camadas que são executadas sobre a camada básica, conforme estas camadas forem sendo definidas como padrão.

O "X-Terminal" implementado para PEGASUS-32x consistia, basicamente, de um processador e memória local executando as funções necessárias para dar suporte ao servidor do sistema "X-Windows". Além disto, possui um monitor colorido de alta resolução, um teclado e um "mouse". Possuía também um processador auxiliar para realizar as tarefas de traçado e exibição gráficas.

A Conexão do "X-Terminal" ao PEGASUS-32x era feita através de uma interface serial RS-232. No PEGASUS-32x/PLURIX estava implantada a biblioteca "X-Lib", permitindo a criação, neste sistema de diversos clientes do sistema "X-Windows". Estes clientes/aplicações exibem os seus resultados no "X-Terminal" recebendo e enviando mensagens através do "X-Protocol" transmitido pela linha serial, que liga o "X-Terminal" ao PEGASUS-32x.

A implementação de um "X-Terminal" tornou viável o rápido desenvolvimento de uma interface gráfica para o PLURIX. Por isto, não se levou em consideração a perda de desempenho ao se utilizar uma interface serial para ligar o "X-Terminal" ao PEGASUS-32x. O objetivo principal era desenvolver uma interface gráfica que possibilitasse a implementação de outras camadas mais sofisticadas. Sob

este aspecto, estudou-se a implantação de bibliotecas de mais alto nível, com por exemplo a MOTIF da Open Software Foundation, sobre a biblioteca “X-Lib” implementada no PLURIX.

Em paralelo, começou a ser definida e implementada no NCE/UFRJ, uma Estação de Trabalho. Esta Estação terá como Sistema Operacional o PLURIX, e pôde aproveitar toda a interface gráfica já desenvolvida. Para esta Estação, o PLURIX estaria com as chamadas ao sistema que permitissem a ligação a estação a redes TCP/IP. Este tipo de ligação ainda não tinha sido implementada na época (ainda estava em desenvolvimento, veja o Capítulo 11).

A partir da implementação da Estação de Trabalho, o “X-Terminal” poderia ser ligado à estação através do Protocolo TCP/IP e de uma interface padrão “ethernet”. Com isto, o desenho dos aplicativos gráficos executados no ambiente Estação de Trabalho/PLURIX e utilizando as bibliiotecas gráficas implantadas sobre o PLURIX, foi consideravelmente maior do que o apresentado no ambiente PEGASUS-32x/PLURIX. A comunicação com “X-Terminal” e consequentemente a velocidade com que os resultados das aplicações são visualizados no “X-Terminal”, seria feita de forma mais rápida através da utilização de uma interface “ethernet” de alta velocidade, ao invés da linha de comunicação serial.

15. A Destruição da Informática Brasileira

A Informática Brasileira está sendo destruída. Depois de um período de construção, a Informática Brasileira está em declínio. As razões deste declínio e as perspectivas para o ano 2000 são os temas polêmicos que gostaríamos de comentar neste capítulo.

Em primeiro lugar, é importante que se defina o que entendemos por Informática Brasileira. A Informática é a capacidade de, a partir de insumos básicos, conceber, projetar, construir, industrializar, comercializar e dar manutenção a sistemas computacionais. Ultimamente a Informática tem sido erroneamente apresentada como o uso de sistemas computacionais. Lembramos que é preciso diferenciar a informática em si do uso da informática. Certamente o uso da informática é importante, assim como bem utilizar um automóvel, por exemplo. Porém, conceber, projetar e construir um veículo é algo bem mais complexo do que seu uso de forma correta e eficiente.

A Informática é qualificada como sendo própria de um determinado país, quando indivíduos com residência no país, detém esta capacidade de, a partir de insumos básicos, conceber, projetar, construir, industrializar, comercializar e dar manutenção a sistemas computacionais. Assim é natural nos referirmos à informática japonesa, americana, alemã, etc ...

A capacidade acima descrita é chamada autonomia tecnológica em informática. Autonomia tecnológica significa, portanto, possuir a capacidade e não, necessariamente, exercê-la em todas as áreas da informática. Em particular, por razões mercadológicas ou outras, pode-se temporariamente abrir mão da industrialização de determinados bens de informática sem, no entanto, perder-se a autonomia tecnológica sobre os mesmos. Estabelece-se, assim, a tão festejada interdependência tecnológica característica das relações comerciais do capitalismo moderno entre os países do

primeiro mundo. Por exemplo, nos últimos anos os E.U.A. têm importado memórias japonesas em lugar de fabricá-las. Os japoneses têm licenciado projetos de microprocessadores em lugar de projetá-los. Há, portanto, nítidamente uma interdependência tecnológica em informática entre os E.U.A. e o Japão, sem que, em nenhum momento, se caracterize que os E.U.A. ou o Japão perderam a sua autonomia tecnológica.

O poder econômico, político e militar conferido aos países capazes de manipular em computadores enorme quantidade de informação e as facilidades irreversíveis introduzidas pelo uso da informática nas mais diversas atividades humanas para resolver problemas específicos de uma determinada sociedade, dão a dimensão da necessidade de um país ter autonomia tecnológica em informática. Em lugar da importação (nem sempre acessível) e adaptação (nem sempre possível) de produtos e serviços, é viável (mais seguro e, a médio prazo, mais barato) concebê-los e produzí-los localmente, orientando-os diretamente para as suas aplicações específicas e/ou para seus usuários finais. A automação bancária no Brasil é um exemplo marcante.

O uso da informática nos mais diversos países continuará crescendo independentemente do fato de a tecnologia de informática ser dominada localmente ou não. O custo econômico e político, refletindo-se invariavelmente no social, deste uso é que será determinado pela autonomia tecnológica do país. Pode-se afirmar, sem medo de errar, que um país que pretenda ser independente no século XXI tem de dominar as tecnologias de que se utiliza e, dentre estas, pela sua abrangência, a tecnologia de informática é essencial.

Enganam-se redondamente aqueles que julgam poder comprar tecnologia. Compram-se produtos e processos, adquire-se “know how”, mas a tecnologia se esconde no “know why”. É sabendo-se por que determinados produtos são concebidos e determinados processos são utilizados, que a inteligência de um país pode conceber e construir os novos produtos e processos de que o país necessita. A tecnologia está na cabeça do profissional. Não se transfere tecnologia desacompanhada do profissional que a detém. Como a transferência de pessoas muitas vezes não é simples, é preciso que a tecnologia seja desenvolvida localmente e que os profissionais que a detêm tenha incentivos para que a utilizem no país.

Tem-se demonstrado ao longo dos anos, nos mais diversos países, que desenvolvimento de tecnologia visando a autonomia tecnológica em áreas ainda não dominadas localmente somente pode ser feito através de um esforço conjunto do governo, da indústria e da universidade. O governo, através de políticas coerentes de natureza fiscal, incentivos diretos, manuseio de seu poder de compra, etc. pode sinalizar às indústrias para investirem em determinados produtos ou processos, diminuindo o risco naturalmente envolvido no desenvolvimento de novas tecnologias. A indústria, gerando produtos baseados em tecnologia nacional e investindo no desenvolvimento de tecnologia própria em áreas que a médio prazo poderão gerar produtos competitivos, mesmo no mercado internacional. A universidade, por sua vez, contribuindo com a formação de mão de obra especializada para viabilizar esta política, além de utilizar a sua mão de obra permanente e flutuante (professores, pesquisadores, alunos, etc.) para a concepção e construção de protótipos utilizando tecnologias ainda não totalmente dominadas no país

Para que este tripé (governo, indústria, universidade) funcione adequadamente é preciso planejamento, no mínimo, de médio prazo. Resultados em desenvolvimento de tecnologias não dominadas em um país somente podem ser claramente percebidos ao longo de décadas e nunca ao

longo de poucos anos. O processo é lento e seus maiores beneficiários são sempre as gerações futuras. O exemplo do Japão e outros tigres asiáticos é eloqüente.

Poucos são os páises hoje no mundo que têm total autonomia tecnológica em informática, mas muito menos ainda são aqueles que não a têm e estão tentando obtê-la. Este tem sido o caso do Brasil nos últimos anos.

Nos anos 70 (do século XX) começamos corretamente. Reservamos uma parte bastante pequena do mercado (computadores médios e pequenos) para ser explorada por fabricantes nacionais. Para ocupar rapidamente o mercado e ainda acreditando que tecnologia pudesse ser transferida, importamos inicialmente projetos de computadores médios estrangeiros enquanto, num esforço conjunto indústria/universidade, desenvolvíamos os nossos próprios. Ao final de alguns anos verificou-se que os projetos nacionais foram muitíssimo mais bem sucedidos no mercado (verifique a linha COBRA 500, por exemplo) do que seus similiares estrangeiros. A criatividade nacional dava mostras do seu potencial. Era preciso preservar e incentivar esta capacidade.

No início do anos 80 surgiram os PCs. Foi um grande golpe na criatividade brasileira. Em lugar de conceber novos produtos, era preciso criar “clones”, sistemas idênticos ao que já existiam no exterior. A política governamental exigiu também que o “hardware” e o “software” de PCs fossem integralmente feitos no país. Como isto era inviável para a grande maioria das empresas, iniciou-se um política de “vista grossa” que perdurou até os anos 90. Muitas indústrias fingiam que projetavam e os órgãos fiscalizadores fingiam que acreditavam. Sem dúvida, as empresas, pricipalmente as que encurtaram o processo de desenvolvimente, ganharam muito dinheiro às custas dos usuários, naturalmente. Além disto, uma das poucas empresas que certamente dedicou-se ao desenvolvimento (verifique o sistema operacional SISNE, da SCOPUS, por exemplo) foi paradoxalmente penalizada e teve mais tarde o seu produto descartado pelos próprios organismos governamentais que exigiram o seu desenvolvimento. A política de informática se desmoralizava e a destruição da informática brasileira começava a se anunciar.

Ainda nos anos 80, começaram a ser desenvolvidos os chamados supermicros. Baseados em arquiteturas abertas e utilizando microprocessadores de 32 bits, os supermicros começaram a ocupar o espaço dos minicomputadores. Profissionais da área mostraram ainda que tais sistemas iriam em breve substituir inclusive os populares superminis da época. Mesmo assim, não contentes com o fracasso da primeira tentativa de importação de projetos estrangeiros e contrariando a recomendação original da política de informática que exigia o desenvolvimento nacional a partir da segunda geração, repetiu-se o processo de importação. Importaram-se projetos em vias de obsolecência e ocuparam-se profissionais competentes na absorção de um conhecimento sem futuro. No entanto, as empresas novamente ganharam dinheiro, sempre às custas dos usuários. Mais uma vez a criatividade brasileira era vilipendiada.

No final dos anos 80, muitas empresas, com o beneplácito do governo tiraram a máscara e passaram a afirmar abertamente que é imprescindível a sua associação com o capital estrangeiro e que o desenvolvimento de soluções nacionais se encontra na “contramão da História”. Mas há de se perguntar: E os usuários e os contribuintes que financiaram esse desenvolvimento para ter em futuro próximo soluções nacionais de qualidade e preço compatíveis com aqueles praticados no exterior, como ficam nesta história? Pagam a conta e ficam quietos esperando que as empresas nacionais acertem desta vez a tecnologia que vão comprar? E os profissionais competentes que

decidiram ficar neste país porque a política de informática sinalizava para a existência de trabalho qualificado e ganhos compatíveis com o trabalho realizado? Mudam de ramo? Vão vender computadores ou pacotes de "software" estrangeiro? Ou mudam de país? Pois eu acho que chega! A política de informática não foi feita para se proteger apenas o capital nacional, embora se reconheça que ele é essencial para a viabilização da tecnologia brasileira de informática. Mas se a capacidade nacional desde a concepção até a manutenção de sistemas computacionais, que definimos como Informática brasileira, não vai ser preservada, a política de reserva de mercado perde o seu sentido. Não é de se estranhar que os usuários, grandes financiadores do processo, passem a reclamar o direito de poder importar bens e serviços de informática sem taxas e sem intermediários. Nem burocratas do governo de um lado, nem "representantes" nacionais do outro.

Infelizmente, a solução de importação é inviável a médio e longo prazo. No caso da liberação das importações, poderemos comprar computadores mais baratos e de melhor qualidade enquanto houver divisas. A inexorável falta de divisas, porém, tornará a importações direta proibitiva e fará com que tenhamos que nos voltar novamente para o produtor estabelecido no país. Nessa época, porém, as empresas nacionais terão passado a ser simples montadoras de multinacionais. O conhecimento que já havíamos adquirido ao longo dos anos terá sido irremediavelmente perdidos, pois as universidades terão deixado de orferecer cursos "exóticos", como arquitetura de computadores e sistemas operacionais, para dedicar-se à formação de bons usuários das sofisticadas máquinas produzidas pela indústria "nacional". Estas empresas "nacionais" certamente estabelecerão centros de pesquisas nas próprias indústrias ou em universidades que estarão ligadas por satélites diretamente às suas matrizes no exterior. Desta forma poderão aproveitar no exterior a capacidade de profissionais formado com recursos brasileiros, mas que não queiram mudar de país nem se adaptar a uma nova função da Universidade na área de Informática. A estes profissionais, entretanto, jamais será dada uma visão de conjunto dos sistemas em que trabalham para descartar qualquer possibilidade de autonomia tecnológica. Aliás, eles não nem perceberão que não tem esta visão de conjunto, pois o uso eficiente da informática estrangeira será alardeado como grande avanço da "tecnologia" de Informática no Brasil e haverá inclusive intelectuais defendendo veentemente esta tese.

Já no início do século XXI sentiremos intensamente os problemas advindos da falta de domínio da tecnologia de informática. Produtos deixarão de ser fabricados, mercados não poderão ser conquistados, a dependência militar se agravará, serviços e equipamentos terão os seus preços determinados no exterior, etc... Mas já não haverá no país a capacidade tecnológica para tentar reverter este quadro. Ela terá sido destruída já há muito para que não ficássemos na "contramão da História". E assim, o Brasil voltará a sua "vocação" de fornecedor de produtos primários e terá perdido mais uma vez a batalha pela autonomia tecnológica nacional. Daqui a algumas gerações, talvez surjam novos personagens demonstrando mais um vez que o ato de 1822 (Independência do Brasil) vai continuar inócuo enquanto o país não dominar a tecnologia de que necessita. Tenho receio de que até lá o mundo haja mudado tanto que a obtenção de tecnologia se torne praticamente impossível pelos países que ainda não a detém. Talvez as novas gerações jamais nos perdoem pela oportunidade que tivemos e desperdiçamos.

16. A Criação do TROPIX

Com o término da Reserva de Mercado (após a Destruição da Informática Brasileira), não mais seriam fabricados PEGASUS-32x nem ÍCARUS para executarem o PLURIX. O que fazer para salvar o PLURIX? Estudando as diversas alternativas, a mais promissora era a utilização do PC. Mas será que o PC (na época ainda com o processador 80386) seria adequado para executar o PLURIX? Felizmente, isto foi discutido e confirmado no artigo "Marrying UNIX and the 80386" de Carl Hensler e Ken Sarno (Byte, abril de 1988).

Felizmente, no NCE ainda havia alguns ÍCARUS funcionando, e neles iniciamos o transporte do PLURIX para os PCs. Foi estabelecido que este Sistema Operacional, depois de transportado seria chamado de TROPIX. O primeiro passo para este transporte foi o de construir um montador ("assembler") para o processador do PC, o INTEL 80386, que é bem diferente do MOTOROLA 680x0. As diferenças básicas são:

1. As instruções de máquina do MOTOROLA 680x0 podem possuir até dois endereços de memória, e possuem instruções de autoincremento/decremento. Isto, que permite uma programação poderosa e compacta, causa grandes problemas no caso de acesso a uma posição de memória inexistente e/ou inacessível ("Bus error"). Isto ocorre, pois é bastante complexo descobrir-se qual dos endereços causou o erro e, se for o caso, desfazer o incremento/decremento do primeiro endereço. Por outro lado, o INTEL 80386 possui no máximo apenas um endereço de memória e não possui intruções de autoincremento/decremento, e o processamento para este caso é muito mais simples.
2. O processador MOTOROLA 680x0 possui 16 registros internos, sendo 8 de dados e 8 de endereços, possibilitando o uso de (em torno de) 10 variáveis em registros (cujo acesso é muito mais rápido do que em memória). Por outro lado, o INTEL 80386 possui apenas 8 registros internos, possibilitando o uso de apenas 3 variáveis em registros. No entanto, no MOTOROLA 680x0, possuindo duas classes de registros, é comum o valor de um registro de dados ser necessitado como um endereço (ou vice-versa), forçando uma cópia.
3. O MOTOROLA 680x0 é um processador natural de 32 bits, enquanto que o INTEL 80386 é um processador misto, isto é, pode funcionar em 16 ou 32 bits. Quando é ligado (ou reinicializado), funciona no modo de 16 bits, e para passarmos para o modo de 32 bits ele deve ser programado, o que é relativamente complexo.
4. O MOTOROLA 680x0 não contém gerência de memória embutido, devendo realizar-se isto com outros circuitos da placa-mãe, enquanto que o o INTEL 80386 já a possui e (felizmente) é adequado para sistemas operacionais estilo UNIX (como é explicado no artigo referenciado acima).
5. O PC possui uma memória em ROM ("Read-only Memory"), que contém uma BIOS ("Basic Input/Output System"), que auxilia a carga de um sistema operacional. Ela contém inúmeras funções, sendo as principais a escrita na tela e leitura/escrita nos discos/disquetes, operando no modo de 16 bits. No entanto, quando o sistema operacional passa para o modo de 32 bits, perde o acesso à BIOS, a não ser que utilize um modo complexo de usar segmentos mistos (de 16 e 32 bits).

Um problema básico para o transporte era a necessidade de um meio comum aos dois computadores, isto é, no ÍCARUS deveria ser gravado um programa em um determinado meio, que depois seria lido e executado no PC. Felizmente, ambos os computadores possuíam disquetes de 5¼.

Como o INTEL 80386 pode funcionar em dois modos (16 ou 32 bits), o montador deveria poder construir as instruções de máquina para estes dois modos. Tendo já uma primeira versão de um montador para o INTEL 80386 em funcionamento, tentamos gerar um programa de carga, que seria gravado em um disquette de 5¼, lido e executado no PC. Este primeiro teste era um programa bastante simples que apenas escreveria mensagens na tela. Este programa, que era naturalmente em 16 bits, usava a BIOS do PC para isto.

Quando o PC é inicializado, ele lê apenas um setor (512 bytes) de um dispositivo previamente escolhido. Como um programa da carga de um sistema operacional provavelmente não cabe em apenas um setor, o próprio programa de carga é responsável pela carga do resto de si próprio ... O teste seguinte seria tentar ler setores do próprio disquete. Isto teria dois objetivos: em primeiro lugar ler o restante do próprio programa de carga, e em seguida (após possível diálogo pelo teclado/vídeo) ler o sistema operacional. No caso do TROPIX, no entanto, o processo de carga é em duas etapas, isto é, o programa de carga não carrega diretamente o sistema operacional, mas sim um estágio seguinte do programa de carga. Estas duas etapas são denominadas de “boot1” e “boot2”.

O programa “boot1” funciona no modo de 16 bits, mas deve conhecer o formato do sistema de arquivos do TROPIX, pois nele tem de achar o “boot2”, que então irá entrar no modo de 32 bits, e ler o sistema operacional do sistemas de arquivos escolhido. Repare que o “boot2” funcionando já em 32 bits, perde o acesso à BIOS, e portanto, para acessar os discos rígidos (ou inicialmente o próprio disquete) necessita de acionadores (“drivers”) próprios.

Para testar o “boot1”, foi gerado um disquete com um sistema de arquivos do TROPIX em um computador ÍCARUS, e nele foi colocado o “boot1” e um protótipo do “boot2” que irá simplesmente colocar uma mensagem na tela indicando que foi carregado.

Uma vez o “boot1” funcionando e conseguindo ler o “boot2”, temos pela frente duas tarefas não triviais: passar para o modo de 32 bits (perdendo o acesso à BIOS, e mesmo assim necessitando de escrever na tela) e obter um compilador “C” para o INTEL 80386. Enquanto que o “boot1” pôde ser inteiramente escrito em linguagem simbólica, o “boot2” que é bem mais complexo, teve de ser escrito em “C”.

Para tanto, a melhor hipótese era a utilizar o compilador GNU (de distribuição pública), que na época ainda estava em seus estágios iniciais. No capítulo seguinte iremos descrever como ele foi obtido. Uma vez tendo o compilador “C”, o “boot2” pôde ser programado, e como no modo de 32 bits perde-se o acesso à BIOS, ele teria de ter acionadores para os principais dispositivos do PC: teclado, vídeo, disquete e discos rígidos IDE PATA (padrão paralelo). Na época, estes dispositivos eram padronizados, isto é, em qualquer PC estes dispositivos eram do mesmo tipo e eram acessados do mesmo modo. Mais tarde, surgiram teclados USB, discos rígidos IDE SATA (padrão serial). Isto sem falar do “mouse”, que na época era serial, e depois surgiram os modelos PS/2 e USB.

Quando o “boot2” já estava em condições de carregar um protótipo do núcleo do TROPIX (“kernel”) do disco para a memória, começamos a nos preocupar com o funcionamento do INTEL 80386, que naturalmente é completamente diferente do MOTOROLA 680x0.

No início do núcleo temos de nos preocupar com diversos detalhes do processador, tais como: obter o tamanho da memória disponível do PC, inicializar o relógio, preparar o vetor e o controlador de interrupções, carregar a gerência de memória e procurar dispositivos presentes no PC. Isto é realizado de dois modos: no modo antigo, procurando em portas específicas do barramento ISA (Industry Standard Architecture) e no modo novo, percorrendo o barramento PCI (Peripheral Component Interconnect).

Uma vez preparado o processador, podemos inicializar as diversas tabelas do núcleo, tais como a tabela de processos, tabela de áreas de entrada/saída, tabela de nós-índices (para acessar os arquivos). Em seguida preparamos artesanalmente o processo 0 (o escalador do sistema operacional, que sempre irá escolher o próximo processo a ser executado) e o processo 1 (que irá executar o programa “init” e ao cabo de várias etapas irá iniciar um interpretador de comandos (“sh”) no vídeo.

A etapa seguinte consistiu em compilar os comandos essenciais, tais como o “init”, “sh” e alguns outros, tais como “cat”, “du”, “df”, etc ... cuja função seria a de testar o funcionamento do sistema. O método então consistiu em gerar um disquete de 5¼ (no ÍCARUS) contendo o “boot1”, “boot2”, núcleo do sistema, os programas citados acima e o executar no PC.

Uma vez com esta fase concluída, o passo seguinte é a geração de um sistema de arquivos do TROPIX no disco do PC. Para tanto foi necessário o programa “mkfs”, para a geração do sistema, e em seguida o programa “cp” para copiar todos os arquivos do disquete para o disco rígido do PC.

Finalmente, quando todos os arquivos estiverem no disco rígido do PC, pode-se inicializar o TROPIX diretamento do PC, e o transporte foi um sucesso!

17. Obtendo o Compilador “C” para o PC

Como já foi mencionado, para o funcionamento do TROPIX em um PC era necessário um compilador “C” rodando no PC e gerando código para o PC. Acredito ser interessante enumerar as diversas etapas para obtê-lo, utilizando um computador ÍCARUS.

O ponto de partida foi a utilização do compilador GNU (de distribuição pública), que na época ainda estava em seus estágios iniciais. Aliás, o “ANSI-C”, uma versão melhorada do “C”, também estava sendo desenvolvido nesta época. O Compilador “C” para o MOTOROLA 680x0 disponível no ÍCARUS não reconhecia o “ANSI-C”, mas felizmente o compilador GNU, que aceita como entrada o “ANSI-C”, era escrito em “C” simples (Não “ANSI”). Compilar o Compilador GNU (que iria gerar código para o INTEL 80386) no compilador do ÍCARUS (para o MOTOROLA 680x0) foi uma tarefa difícil, que ao final deu certo. Um detalhe que ainda teve de ser feito é a adaptação da saída do compilador GNU (que é a linguagem simbólica para o INTEL 80386) para a entrada do montador

recém construído: o nomes dos registros, nomes das instruções, ordem dos operandos, etc. eram bem diferentes.

Tínhamos o compilador "C" do PLURIX, rodando no ÍCARUS (o mesmo do PEGASUS-32x) e o código fonte (público) do Compilador "C" do GNU, que é paramétrico, contendo tabelas para possibilitar a geração do código objeto para diversos processadores. As etapas foram:

1. No computador ÍCARUS, compilar o código fonte do GNU com o compilador do ÍCARUS. O compilador obtido (que chamaremos de "GNU A"), roda no ÍCARUS e gera programas objetos para o PC (isto é, para o INTEL 80386).
2. No computador ÍCARUS, compilar o código fonte do GNU com o compilador "GNU A". O compilador obtido (que chamaremos de "GNU B"), irá rodar no PC e gera programas objetos para o PC. Repare que compilamos o compilador com o próprio compilador! Isto é um dos momentos da Informática que parece surrealista ...
3. No PC, compilar o código fonte do GNU com o compilador "GNU B". O compilador obtido (que chamaremos de "GNU C"), também roda no PC e gera programas objetos para o PC. A etapa 3 parece supérflua, no entanto é o melhor teste do próprio compilador! Se os compiladores "B" e "C" forem idênticos, é sinal de que o processo de criação do compilador "C" para o PC foi um sucesso.

18. Aprimoramentos e Atualizações do TROPIX

Uma vez com o TROPIX já rodando no PC, foram realizadas muitos aprimoramentos/atualizações no TROPIX, ao longo dos anos. Vamos inicialmente dar uma lista das modificações do núcleo do TROPIX:

1. Novembro de 1990: Criados os serviços de Eventos, Semáforos e Memória Compartilhada a nível de usuário.
2. 1992-1993: Implementada a INTERNET.
3. 1995: Elaborado um acionador para o controlador SCSI 1542 da firma "Adaptec".
4. Agosto de 1996: Implantação de um protetor de tela (para o modo de texto).
5. Janeiro de 1997: Criação de uma primeira versão de um acionador para o ZIP, versão paralela (o ZIP é um disco magnético/ótico removível de 100 MB).
6. Fevereiro de 1997: Início do funcionamento do "X-Windows" no TROPIX.
7. Junho de 1999: Programado um acionador para a placa de som "Sound Blaster 16" da firma "Creative".
8. Dezembro de 1999: Criação das bibliotecas compartilhadas. Deste modo as bibliotecas são compartilhadas por todos os programas executáveis, e com isto estes ficam muito menores.
9. Maio de 2000: Realizado o acionador para a placa "ethernet" de 100 Mbs da firma "RealTek". Com isto, a comunicação através da INTERNET aumentou significativamente de velocidade.
10. Setembro de 2001: Programado um acionador para a placa SCSI 29160 da firma "Adaptec".
11. Dezembro de 2001: Criada a camada de Nós-Índices virtuais, o que possibilitou a montagem de outros sistemas de arquivos (além do nativo do TROPIX).

12. Janeiro de 2002: Elaboração da montagem de sistemas de arquivos FAT12/16/32 do MS-DOS e Windows.
13. Março de 2002: Montagem de discos removíveis CD-ROM.
14. Abril de 2002: Criação de elos simbólicos (além dos elos físicos do UNIX original).
15. Agosto de 2002: Elaboração de um novo sistema de arquivos para o TROPIX denominado “T1”, com 4 KB de bloco lógico ao invés de 512 bytes.
16. Outubro de 2002: Adicionado o sistema de arquivos EXT2 do LINUX, inicialmente somente para leituras.
17. Novembro de 2002: Concluído o sistema de arquivos EXT2 do LINUX com as escritas.
18. Novembro de 2003: Início de funcionamento do acionador de dispositivos USB, inicialmente com o protocolo 1.1, com o “mouse” funcionando.
19. Janeiro de 2004: Melhorias no acionador USB: Controladores OHCI e EHCI.
20. Julho de 2004: Início da elaboração da montagem de sistemas de arquivos NTFS, somente para leituras.
21. Agosto de 2004: Montagem de imagens de sistemas de arquivos (“pseudo-dispositivos”).
22. Setembro de 2004: Começou a funcionar a leitura/escrita de discos removíveis USB.
23. Agosto de 2005: Início da montagem de sistemas de arquivos remotos , através do protocolo NFS versão 2.0.
24. Julho de 2006: Implementação do modo de texto SVGA (1024 x 768 pixels).
25. Agosto de 2006: Versão preliminar do protocolo DHCP para a Internet.

19. O Transporte do X-Window para o TROPIX

Uma vez com o TROPIX funcionando no modo texto, sem dúvida, era imprescindível obter uma interface gráfica. O transporte do Sistema “X-Window” para o TROPIX foi realizado pelo meu colega Oswaldo Vernet de Souza Pires, e foi iniciado em tôrno de 1995. As várias versões do código fonte do sistema iam sendo obtidas a partir da página oficial do sistema, “www.x.org”, mas houveram várias dificuldades para este transporte. Entre elas, citamos:

1. Entender como os arquivos eram compilados, e construir os “Makefiles” compatíveis com o TROPIX. Nas primeiras versões, foi consultada a compilação do Sistema “X” em um outro Sistema UNIX para entender a ordem das compilações e as opções utilizadas nos comandos de compilação. Esta tarefa (que à primeira vista deveria ser simples), levou vários meses.
2. Compatibilizar as chamadas à rede INTERNET usadas pelo sistema “X”, que são baseadas no padrão BSD (“Berkeley Software Distribution Sockets Programming Interface”), com as chamadas do TROPIX, que são baseadas na biblioteca XTI (“X/Open Transport Interface”) .
3. Compatibilizar a chamada ao sistema “select” do BSD com a chamada “attention” do TROPIX.
4. Um detalhe que causou grande dificuldade (que só foi descoberto após um tempo enorme de depuração) foi a suposição, por parte do Sistema “X”, de que as funções de cópia de memória a memória em linguagem simbólica restauravam o sentido da cópia (de baixo para cima ou vice-versa), o que não estava sendo feito no TROPIX.

No final de 1997 o “servidor X” começou a funcionar, desenhando o seu característico desenho de fundo de tela. As etapas seguintes foram a compilação da “xlib” (a biblioteca de funções do Sistema “X” e a compilação de alguns clientes “X”, tais como “fvwm” (o gerenciador de janelas), “xterm” (um emulador de terminal de vídeo), “xclock” (uma imagem de relógio e data)”. Mas tarde foram elaborados outros clientes: “xcpu” (monitoramento do uso da UCP), “xdefrag” (desfragmentação de díscos rígidos), “xfm” (gerenciador de arquivos), “xmandel” (construção de fractais segundo a fórmula de Mandelbrot) e “xview” (visualização de imagens).

Ao total, foram transportadas 5 ou 6 versões do Sistema “X”, normalmente feitas para podermos rodar o TROPIX com uma nova placa de vídeo. No final, bastavam poucas horas para cada transporte.

20. Agradecimentos e Referências

Diversos artigos e publicações foram consultados para a confecção deste histórico. Sem dúvida, os principais deles foram diversas edições do “Boletim do PLURIX”, o qual foi editado pelo NCE/UFRJ durante os anos de 1987 a 1991. Foram editados/reescritos artigos de Newton Faller, Alexandre J. Beltrão Moura, Luiz Fernando Huet de Bacellar, Gabriel Pereira da Silva, Oswaldo Vernet de Souza Pires e Pedro Salenbauch.

Outro artigo consultado foi “Técnicas de Projeto utilizados na Construção do Supermicrocomputador PEGASUS 32-x e do Sistema Operacional PLURIX”, de Newton Faller, Manuel Lois Anido e Pedro Salenbauch, VI Congresso Chileno de Engenharia Elétrica, Santiago, Chile, Novembro de 1985.

Outros artigos publicados sobre o TROPIX:

1. “UFRJ começa a comercializar ‘software’ (denominado PLURIX) feito por professores” , O Globo, 11 de setembro de 1988.
2. “Sistema multitarefa construído em clima tropical”, O Globo (caderno de informática), 8 de fevereiro de 1999.
3. “TROPIX, o nosso mini LINUX?”, Paulo C. Barreto, Info, Julho de 1999.
4. “Mutiusuário e multitarefa: O TROPIX, Sistema Operacional criado pelo NCE/UFRJ está disponível para uso”, Jornal do Brasil (caderno de informática), 12 de agosto de 1999.
5. “TROPIX, uma ferramenta de trabalho que faciilta a vida de alunos de mestrado e graduação”, Roberto Duarte, Revista Canal Médico, Novembro de 1999.